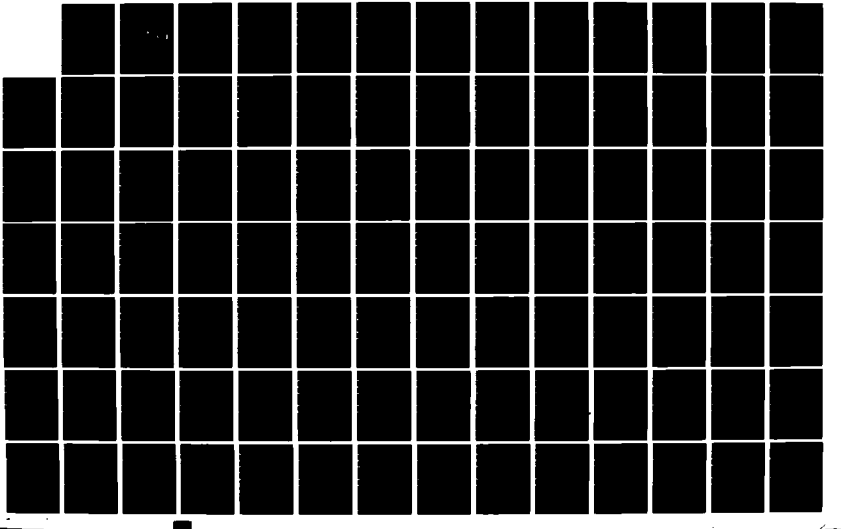
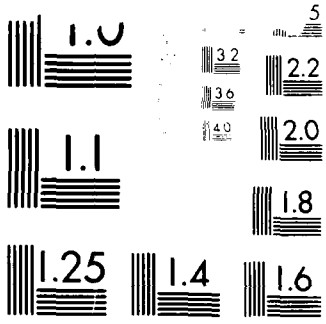
AD-A168 843 A MECHANIZED DECISION SUPPORT SYSTEM FOR ACADEMIC SCHEDULING(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S M FENSTERMACHER MAR 86 1/3

UNCLASSIFIED F/G 5/9 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
3.2
3.6
4.0
5

AD-A168 843

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
ELECTE
JUN 27 1986
S D
D

THESIS

A MECHANIZED DECISION SUPPORT SYSTEM
FOR
ACADEMIC SCHEDULING

by

Stephen M. Fenstermacher

March 1986

Thesis Advisor: Norman R. Lyons

Approved for public release; distribution is unlimited.

DTIC FILE COPY

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): 54	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO.

11 TITLE (Include Security Classification)

A MECHANIZED DECISION SUPPORT SYSTEM FOR ACADEMIC SCHEDULING

12 PERSONAL AUTHOR(S)

Fenstermacher, Stephen M.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1986 March	193

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Scheduling, Decision Support System, Software Design

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis documents the mechanization of Computer Systems Management course selection and scheduling. It examines the factors that complicate academic scheduling and reviews the major problems associated with the current system. Alternative decision support system techniques to consolidate and integrate scheduling data were examined for potential implementation problems. Solutions to those problems became design objectives. The scheduling software was designed using the software engineering methodology. A prototype program was distributed to selected users and the final program was modified based on their feedback. The program custodian should control the program by making it available through a centralized media such as an electronic bulletin board. Future enhancements to this program should include expansion of the database, conversion to a database language and development of a microcomputer version.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL: Norman R. Lyons	22b TELEPHONE (Include Area Code): (408) 646-2666 / 22c OFFICE SYMBOL: 54Ly

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted; All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

A Mechanized Decision Support System
for
Academic Scheduling

by

Stephen M. Fenstermacher
Captain, United States Marine Corps
B.S., The Pennsylvania State University, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1986

Author: _____
Stephen M. Fenstermacher

Approved by: _____
N.R. Lyons, Thesis Advisor

_____
M.P. Spencer, Second Reader

_____
W.R. Greer, Jr., Chairman,
Department of Administrative Sciences

_____
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

This thesis documents the mechanization of Computer Systems Management course selection and scheduling. It examines the factors that complicate academic scheduling and reviews the major problems associated with the current system.

Alternative decision support system techniques to consolidate and integrate scheduling data were examined for potential implementation problems. Solutions to those problems became design objectives. The scheduling software was developed using the software engineering methodology. A prototype program was distributed to selected users and the final program was modified based on their feedback.

The program custodian should control the program by making it available through a centralized media such as an electronic bulletin board. Future enhancements to this program should include expansion of the database, conversion to a database language and development of a microcomputer version.

TABLE OF CONTENTS

I. INTRODUCTION 6
A. BACKGROUND 6
B. SCHEDULING 7
C. THESIS OUTLINE 9

II. DECISION SUPPORT SYSTEMS 10

III. SOFTWARE DESIGN 13
A. GENERAL 13
B. PLANNING PHASE 14
C. DEVELOPMENT PHASE 14
D. MAINTENANCE PHASE 15

IV. IMPLEMENTATION, FEEDBACK & MODIFICATIONS 17
A. PROTOTYPE RELEASE 17
B. INITIAL IMPLEMENTATION 17
C. PROGRAM REVIEW 18

V. CONCLUSIONS AND RECOMMENDATIONS 19
A. CONCLUSIONS 19
1. Software Engineering 19
2. The Skeduler Program 19
B. RECOMMENDATIONS 20
1. Software Engineering 20
2. The Skeduler Program 20

APPENDIX A: EDUCATIONAL SKILL REQUIREMENTS FOR COMPUTER SYSTEM SUBSPECIALTIES 22

APPENDIX B: THE SOFTWARE PLAN 24

APPENDIX C: DATA FLOW DIAGRAM 28

APPENDIX D: DATA DICTIONARY 39

APPENDIX E: USER'S GUIDE 55

APPENDIX F: STRUCTURE CHART 81

APPENDIX G: MODULE DESCRIPTIONS 91

APPENDIX H: PROGRAM LISTING 120

APPENDIX I: INTEGRATION TEST SPECIFICATION 176

APPENDIX J: MAINTENANCE MANUAL 182

LIST OF REFERENCES 189

BIBLIOGRAPHY 190

INITIAL DISTRIBUTION LIST 192

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

I. INTRODUCTION

A. BACKGROUND

The mission of the Naval Postgraduate School (NPS) is "to conduct and direct the advanced education of commissioned officers, and to provide other technical instruction to meet the needs of the naval service." Consequently, the Superintendent confers Bachelor's, Master's, Engineer's or Doctor's degrees on qualified graduates. The Superintendent's authority is subject to Navy regulations and is contingent on curricula accreditation.

The school's unique mission requires that curricula design be approved by both the academic community and the military/user community. The academic community sanctifies the educational structure and content through curricula accreditation.

The school meets military/user requirements for curricula design through an evolutionary process. NPS faculty and staff are in continual contact with curriculum sponsors and end users. Sponsors include such organizations as Naval Data Automation Command, Naval Supply Systems Command, Naval Telecommunications Command, and United States Marine Corps Headquarters. The NPS faculty and staff consolidate the information derived from these contacts and generate curriculum skill requirements. (see Appendix A) The Naval Postgraduate School conducts a bi-annual curriculum review to evaluate the relationship between the available courses and the skill requirements. The final products of this review are the matrix of required courses, a list of approved electives, and the design of alternative emphasis area structures for the curriculum in question.

The academic departments use the matrix of required courses to schedule classes for all new incoming students.

This provides a standard instructional package within each curriculum. Within the CSM curriculum, there are only two instances where students may modify or add to the standard matrix. The first is through course validation. If students pass validation examinations, they are exempt from having to take the validated course. Nevertheless, they must schedule another course in its place. The validated course counts as a prerequisite for future courses and towards all other graduation requirements.

The second instance when students may deviate from the standard course matrix is through the selection of emphasis area courses. At the beginning of their second term, all CSM students must choose one of four CSM emphasis areas. They must then select and schedule courses that are encompassed in that area. The selected courses must be tentatively scheduled to fill open matrix slots in the next four academic quarters.

B. SCHEDULING

It is difficult to make realistic scheduling projections for one overriding reason. The Naval Postgraduate School attempts to let student desires drive the selection of courses offered. Therefor, the academic departments ask students to project a schedule of desired courses. Students are often notified that the school will not offer their selected courses. Then the selection/scheduling process begins again.

The student is free to choose from any discipline in scheduling a course for replacement of one validated. By scheduling wisely, the student can use this opportunity to meet prerequisites to expand future course selection options. Wise scheduling requires research.

In scheduling courses for emphasis area requirements, the student must consider several things. Does the course fulfill the appropriate emphasis area requirement? Has the

student met all prerequisites for the selected course? Will the school teach the course in the desired academic quarter? Does the course fulfill other graduation requirements? Answering these questions also requires research.

Where can the student go for the answers to these questions? The "course catalog" contains information about the course title, description, department, hours and prerequisites. The "tentative course offerings manual" (and I emphasize tentative) tells the student when the school expects to teach the course. CSM curriculum Memo of 25 June 1985 outlines emphasis area descriptions and course options. Finally, periodic memos from department chairmen notify students of actual course offering times.

Interviews with CSM students (PL-51 & PL-53) reinforced my contention that the scheduling procedure has several major flaws.

1. the process contains too many different sources for data research.
2. the process is more time consuming than necessary.
3. the process is more complicated than necessary.
4. there is insufficient advanced information provided to make sound decisions.

There are several potential solutions to these problems.

1. cease course validation.
2. establish a standard matrix of required courses per emphasis area.
3. departments could conduct more advanced scheduling of courses.
4. consolidate all required research data into one source.
5. mechanize the entire scheduling process.

The first two solutions are unacceptable according to NPS policy. Academic departments must allow students to validate courses to prevent educational repetition and to exploit individual talents and interests.

The third potential solution is desirable but is not always achieveable. Flexibility is required as a result of students' changing desires. Furthermore, this is a partial solution at best. The school could implement the fourth possibility via manual or mechanized media and thereby help students tremendously. Again, this is a partial solution.

The fifth possibility holds the most promise in that it would incorporate #3 and #4 and could go a step further. It would combine the raw research data with manipulated data and put that information in a format to aid decision making. And it could put that information at the student's fingertips thereby reducing research time and complexity. This thesis will pursue the mechanization solution.

C. THESIS OUTLINE

Chapter 2 will examine the characteristics of decision support systems to identify desirable features for a mechanized scheduling system. It will also examine software design techniques to identify the most appropriate methodology for this problem.

Chapter 3 will discuss the selected design approach in detail. It will address the systematic application of design techniques through each step in the software development process.

Chapter 4 will address the implementation of the completed software package and will outline the feedback received from users. This chapter will also indicate program modifications made as a result of the feedback.

Chapter 5 will draw conclusions and provide recommendations regarding academic scheduling and software design.

II. DECISION SUPPORT SYSTEMS

Keen and Wagner [Ref. 1] define a decision support system (DSS) as a computer-based system which managers use personally on an on-going basis in direct support of their managerial activities. They go on to discuss the capabilities that a DSS must have. It must be able to reflect the way a manager thinks. It must be flexible and adaptive through ease of modification. It must support managers in a complex process of exploration and learning. Finally, it must evolve to meet changing needs, knowledge and situations. The goal of a DSS is to provide the user with a tool that meshes with the users own decision making process. The means to achieve that goal is whatever software and hardware tools are suitable and available.

These desirable capabilities translate into some specific design criteria.

1. flexibility.
2. architecture that allows quick and easy extensions and alterations.
3. interface that buffers the user from the computer.
4. communicative display devices (interactive).

With any system there is a tradeoff between power and ease of use. The proposed scheduling system will have a tight structure that is easy to use in solving specific problems. This will limit the system flexibility but will enhance the system friendliness. This is an acceptable tradeoff. Students will use this program for less than one year. Therefor the program must be easy to use but will not require the high degree of flexibility needed in on-going use Decision Support Systems. Nevertheless, adaptability will be a development consideration and will be achieved through structured design and extensive documentation.

Steven Alter [Ref. 2] places mechanized decision support systems in seven categories:

	NAME	PURPOSE	ORIENTATION
1)	File Drawer	data retrieval	data
2)	Data Analysis	retrieval/analysis	data
3)	Analysis Information	data analysis	data
4)	Accounting Models	simulation	model
5)	Representational	simulation	model
6)	Optimization Models	suggestion	model
7)	Suggestion Models	suggestion	model

File drawer systems provide on-line access to particular data items whereas data analysis systems analyze data files. The problem of mechanizing a scheduling system falls between these two categories. Scheduling requires a large database of information for review and manipulation during decision making. Because of this manipulation, I position the solution model in the data analysis group.

Data analysis systems can be further subdivided into tailored and generalized categories. Tailored systems provide specific analysis functions for definite tasks. Generalized systems provide the ability to perform wide ranging database analysis and produce simple models. Although mechanization of scheduling will provide general review of the database, the solution model more appropriatly falls into the tailored category because of its narrow functional scope.

Alter lists several problems associated with development and implementation of data analysis systems. I consider three to be of critical importance. The first is the unfreezing of job image and the way of approaching problems. All people concerned with current scheduling practices have become accustomed to its manual orientation. Furthermore, most new students are intimidated by the computer. This

must be overcome if a mechanized system is to be effective. The system must be user friendly, it must meet all scheduling needs and it must provide more information than was previously available.

The second area of concern is technical in nature but relates directly to the first problem: retrieval flexibility. The system must be capable of showing a broad range of data but it must also provide a measure of power in manipulating that data. The system must be flexible enough to allow the user to easily perform realistic operations.

The third area of concern deals with user motivation and training. Can potential users figure out what to do with the system? This can be assured with training and strategic timing of system introduction.

A fourth area of concern, not discussed by Alter, is system maintenance. Currency of data is critical. If the system does not provide up-to-date information to users, the students will quickly discard the system. This can be overcome by making the system easily maintainable through proper design, extensive documentation and by limiting the factors that will require change.

The scheduling problems can be solved through mechanization. Nevertheless, the system design must deal with the potential problems discussed above to improve the likelihood of successful implementation.

III. SOFTWARE DESIGN

A. GENERAL

Roger Pressman [Ref. 3] discusses several problems associated with software development. He contends that poor up-front definition is the major cause of software development failure. A detailed statement of function, performance, interfaces, design constraints and validation criteria is critical. According to Deutsch [Ref. 4], early specification of software design objectives increases the overall effectiveness and efficiency of the programming function. Furthermore, it not only reduces the amount of changes required but it identifies the changes earlier in the software lifecycle when they are less costly and disruptive. Pressman also contends that documentation forms the foundation for successful development and provides guidance for effective software maintenance.

Pressman recommends the software engineering methodology as an application-independent technique for software implementation of a problem solution. The objective of software engineering is to establish a set of software components that document each step in the lifecycle. This methodology provides a set of milestones that designers can review at regular intervals throughout the software lifecycle.

The software lifecycle consists of three phases: planning, development, and maintenance. Software engineering addresses specific tasks that designers must accomplish during these phases. During the planning phase, the developer defines the scope of the problem, predicts resource requirements, establishes cost and schedule estimates, and analyzes and defines performance requirements. During the development phase, the developer translates the requirements into software through design, code and test techniques.

Finally during the maintenance phase, appropriate systems personnel take action to correct and prevent software errors, to adapt the software to its changing environment and to perfect the funtional capability of the software.

This thesis will follow the software engineering methodology to accomplish the implementation of the NPS scheduling solution.

B. PLANNING PHASE

Software planning is the first step in the planning phase of the software engineering process. It combines research and estimation to provide the developer with an understanding of the scope of the work, the required resources, the expected cost and effort, [Ref. 5] and the proposed schedule. The Software Plan covers these topics in detail. (see Appendix B)

The next step in the planning phase is Requirements Analysis. This step provides a foundation for software development by describing the flow and structure of information. It also identifies interface details, design constraints and validation requirements. These topics are discussed in detail in the Software Requirements Specification which consists of a data flow diagram (Appendix C), a data dictionary (Appendix D), and a preliminary users guide (Appendix E).

C. DEVELOPMENT PHASE

The development phase translates the set of requirements into an operational system called software. The first step in the development phase is Design. Designers destribute software control by factoring the Data Flow Diagram. They partition it into reception paths, transaction centers and action paths. Then they evaluate each action path for its individual flow characteristics. Next, the designers map the Data Flow Diagram into a software structure amenable to

transaction processing. They develop a modular structure, define interfaces, and establisha data structure. Design documentation consists of the structure chart (Appendix F) and the related module descriptions (Appendix G).

The second step in the development phase is Coding. Programmers translate the module specifications into an appropriate progamming language. A Program Listing displays the software code (see Appendix H).

Testing is the final step in the development phase. It consists of three parts: unit testing, integration testing, and validation testing. Unit testing attempts to validate the functional performance of an individual modular component of software. Integration testing looks at functions and interfaces to provide a means of assembling the software modules. Validation testing ensures that the program meets all software requirements. Top down/incremental integration combines these three parts into one testing method. The Integration Test Specification discusses this testing methodology in detail (Appendix I).

D. MAINTENANCE PHASE

Maintenance is the final phase of the software lifecycle. A goal of the software engineering methodology is to reduce the amount of effort that will have to be expended on maintenance. Developers accomplish this by designing a program for change and by having the planning and design documentation readily available.

Designing a program for change includes several things. Documentation must be complete and understanable. Module cohesion must be high to ensure relative functional strength. Module coupling must be low to ensure relative functional independence.

To aid in program maintenance, we have developed a seperate maintenance manual (see Appendix J). This manual will direct the program maintainer in making simple but

necessary updates. The total compliment of design documentation will be a valuable tool in keeping the SKEDULER program operational and up-to-date.

IV. IMPLEMENTATION, FEEDBACK & MODIFICATIONS

A. PROTOTYPE RELEASE

During june 1985 we sent preliminary copies of the SKEDULER program to ten potential users for review and comment. These users included 3 professors, 1 curricular officer, and 6 CSM students. As a result of their comments, we modified the program in the following areas:

1. menu formatting.
2. information display content & formatting.
3. screen paging.
4. addition of course names.
5. add the ability to overlay a course within the Add_A_Course module instead of having to go to the Drop_A_Course module.
6. add the ability to add a course even when the data base shows it is not offered.
7. improve the Guidance module (more data).
8. show courses validated at the end of the validate procedure as well as during the schedule update.

B. INITIAL IMPLEMENTATION

On 15 July 1985 we sent the SKEDULER program to 26 students in the PL-53 CSM class for use in the selection of emphasis area courses. We also sent it to another professor and a PL-51 student for review and comment. The feedback from the users indicated a time savings of 25 to 50% over the manual system of course scheduling. Nevertheless, they were not completely satisfied with the program. The combined feedback from the users and reviewers resulted in the following program changes:

1. addition of course descriptions
2. reformatting of the updated schedule display
3. addition of the ability to list and add courses from within the emphasis area selection module instead of

having to return to the main menu and select those options seperately

C. PROGRAM REVIEW

On 16 October 1985 review of the structure chart revealed several module cohesion and coupling design flaws. To correct those problems, we made the following program changes:

1. We removed a seperate module to convert lower case input to upper case input We then inserted the conversion function directly in the appropriate modules (i.e., Add_A_Course, Drop_A_Course, Validate, Listind, Describe). This change simultaneously reduced module coupling and increased module cohesion.

2. We removed a seperate module to check the status of required course scheduling. We then inserted the status check function directly in the Update_Show module. Again, this improved both cohesion and coupling characteristics.

3. We inserted the Pick_A_New_Term module into the Add_A_Course module. This was done because the Add module was the sole user of the Pick module.

4. We deleted the Show_Validated_Courses module and inserted its function directly into the Validate and Update_Show modules. This was done to improve both cohesion and coupling.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

1. Software Engineering

The software engineering approach is extremely valuable and effective. This systematic development methodology improves system quality and maintainability and enhances program control. It requires up-front planning that pays off over the life of the program by reducing and simplifying maintenance.

2. The Skeduler Program

Writing the SKEDULER program in a database language would have made both the code and the data easier to maintain. Programming inexperience prevented that approach.

User feedback has shown that the consolidation of information provided by SKEDULER is useful in academic scheduling. Users estimate a time savings of up to 50% over the manual system.

The key to the success of SKEDULER will be its maintenance and continual update. Estimated maintenance effort is only 8 hours per academic quarter (see Appendix B). Neglecting that effort will quickly render SKEDULER obsolete.

Users were disappointed in not being able to add courses that were not listed in the data base. The courses resident in SKEDULER include required courses, emphasis area courses, and all prerequisites for required and emphasis area courses. These catagories will suffice for the majority of students however many students need more options especially if they have validated several required courses. The problem with adding courses not listed in the data base is that a complete set of information related to the course must be resident in the data base to make all SKEDULER

functions completely operational. Nevertheless, this problem must be overcome to make SKEDULER more usable.

Limitations in PASCAL as well as in terminal orientation cause formatting problems when adding new information to the data base. This problem can be overcome by carefully following maintenance instructions during data base update. (see Appendix J)

B. RECOMMENDATIONS

1. Software Engineering

We recommend this methodology for all programs with other than a single use requirement.

2. The Skeduler Program

Implementation will be most effective if the curricular officer posts a precompiled read-only copy of the SKEDULER program on an electronic bulletin board. All CSM system users will have access the bulletin board. This will save space on the users A-disk (the program uses 257K bytes of storage or 15.4% of an A-disk) without reducing access to the program. This will also centralize the program for maintenance and update.

Early maintenance of the program should include an increase in the number of courses listed in the data base. This will eliminate the need for adding courses not listed although it will require more disk storage space. Additions should include the most popular courses from the CS, IS and MN catagories.

Regular program maintenance according to published instructions is essential. This must be done to prevent program obsolesence.

Future adaptations of this program should include conversion to a database language. Furthermore, it might be useful to develop a microcomputer version of the program.

Finally, the SKEDULER program could be integrated with the NPS schedule control system so that SKEDULER output could be electronically transmitted to the Registrar for use in developing the NPS master schedule. Collection and integration of data could be accomplished via spooling.

APPENDIX A

EDUCATIONAL SKILL REQUIREMENTS FOR COMPUTER SYSTEM SUBSPECIALTIES

All officers with advanced degree education in computer systems must possess skills and competencies in Automated Data Processing (ADP) management, computer systems hardware and software, financial and resource allocation management, manpower-personnel management, ADP acquisition management, organization design, and systems analysis and design. Furthermore, the officers must be able to apply these skills to the development, management, and utilization of Department of Defense information systems. These skills and competencies are detailed below.

1. The officer must have a thorough knowledge of ADP management theory and practice to include:

a. ADP feasibility studies including economic analysis, auditing, and the selection, evaluation, acquisition, installation and effective utilization of ADP hardware and software.

b. Computer center facilities planning, production planning and control, system performance evaluation, manpower requirements determination, personnel management, budgeting and financial control, organization design, and security.

c. The process of Navy Department, Defense Department, and congressional decision making on Automated Data Processing matters.

d. The ability to plan and implement a major programming project.

e. The planning, controlling and directing of scarce resources and multiple, time-constrained goals while confronted with high technological change.

2. The officer must have a thorough knowledge of computer hardware and software to include:

a. Basic components of a computer system and their patterns of configuration and communication.

b. An ability to program in COBOL a defined problem and provide the necessary documentation of that program along with an associated minimum knowledge of FORTRAN or PASCAL.

c. The characteristics, effectiveness and system economics of various Navy, Department of Defense and civilian computer-communication networks and services, local networks, and distributed computer systems including real-time considerations.

d. The use of computer-aided software development, documentation and testing facilities.

e. An understanding of technological change, its forecasting and impact on policy and practice.

3. The officer must have skills and competencies related to information systems including:

a. The ability to perform a systems analysis and design for moderately complicated information systems and in this process to utilize a system or software development methodology.

b. The ability to obtain compatibility between information system and organizational design.

c. The ability to use the information system as the integrating tool to provide decision support systems for Navy and Department of Defense management.

d. The ability to distinguish requirements for computer-based versus non computer-based information system applications.

e. The ability to define and implement physical as well as data security and privacy measures for a computer-installation.

f. The ability to apply database systems or file processing systems as required by application.

g. The ability to organize and manage a post-installation computer performance monitoring evaluation optimization program.

APPENDIX B

THE SOFTWARE PLAN

1.0 SCOPE

1.1 PROJECT OBJECTIVE: Create an interactive, terminal oriented, menu driven program that will allow CSM students to modify their academic schedules with minimal need to consult hard copy documentation.

1.2 MAJOR FUNCTIONS

1.2.1 Main Menu Transaction Dispatcher

1.2.2 Prerequisite Check

1.2.3 Validate Courses

1.2.4 Add Courses

1.2.5 Drop Courses

1.2.6 Select An Emphasis Area

1.2.7 List Courses

1.2.8 Update Schedule

1.2.9 Minimum Requirement Fulfillment Status

1.3 OTHER CHARACTERISTICS

1.3.1 Interface: The software must be developed to be compiled and executed on the NPS IBM 3033.

1.4 DEVELOPMENT SCENARIO

1.4.1 Develop the Software Plan

1.4.2 Prepare the Preliminary Users Guide

1.4.3 Develop Dataflow Diagram

1.4.4 Prepare Data Dictionary

1.4.5 Develop Structure Chart (Constantine method)

1.4.6 Develop Module Specifications

1.4.7 Code

1.4.8 Develop Test Specifications

1.4.9 Prepare Maintenance Manual

2.0 RESOURCES

2.1 HUMAN RESOURCES

2.1.1 Consultation

2.1.1.1 Prof Lyons

2.1.1.2 Prof Spencer

2.1.1.3 Prof Schneidewind

2.1.1.4 Cmdr Anderson

2.1.1.5 Cmdr Rautenberg

2.1.2 Design & Programming

2.1.2.1 Capt S. M. Fenstermacher

2.2 HARDWARE RESOURCES

2.2.1 Development System: IBM 3033 accessed via IBM 3278

2.2.2 Target Machine : same as above

2.2.3 Usage Requirements: intermittent development usage

2.2.4 Limitations : 1 megabyte of storage

2.3 SOFTWARE RESOURCES

2.3.1 Support

2.3.1.1 VM/CMS

2.3.1.2 Waterloo Pascal

2.3.1.3 XEDIT

2.3.1.4 Waterloo Script

2.3.2 Utility

2.3.2.1 Personal library of functions and procedures

2.3.2.2 NPS Catalog

2.3.2.3 Tentative Course Schedule

2.3.2.4 Skill Requirements

2.4 AVAILABILITY WINDOWS (no restrictions anticipated)

3.0 COST/EFFORT (using BASIC COCOMO Model)

3.1 GENERAL - The Constructive Cost Model (COCOMO) is designed for quick, early, rough order of magnitude estimates of software cost, effort, and schedule. There are three levels of the COCOMO model: basic, intermediate and detailed. The basic model is applicable to small-to-medium size products developed in a familiar in-house development environment. Because these are the characteristics of the SKEDULER project, the Basic COCOMO Model will be used.

3.2 COMPUTATIONS

3.2.1 Assumptions

3.2.1.1 Mode = Organic

3.2.1.2 KDSI = 3000 (program = 2500 / database = 500)

3.2.2 Development Effort <(MM)d>

3.2.2.1 Basic Equation: (MM)d = 2.4(KDSI)**1.05

3.2.2.2 "SKEDULER" : (MM)d = 2.4(3)**1.05
= 2.4(3.1694019)
= 7.6065646 = 7.6 MM

3.2.3 Annual Software Maintenance <(MM)am>

3.2.3.1 Basic Equation: (MM)am = 1.0(ACT)(MM)d

3.2.3.2 ACT (Annual Change Traffic): The fraction of the software product's source instructions which undergo change during a (typical) year, either addition or modification.
<(DSI added + DSI modified)/ Total Project DSI>

3.2.3.3 Annual Maintenance Effort for "SKEDULER"-

		DSI	
Database:	Additions	20	
	Modifications	50	
Program :	Additions	0	
	Modifications	10	
		80	TOTAL

ACT = 80/3000 = .0266667 = .027

(MM)am = 1.0(ACT)(MM)d = 1(.027)(7.6 MM) = .2052 MM

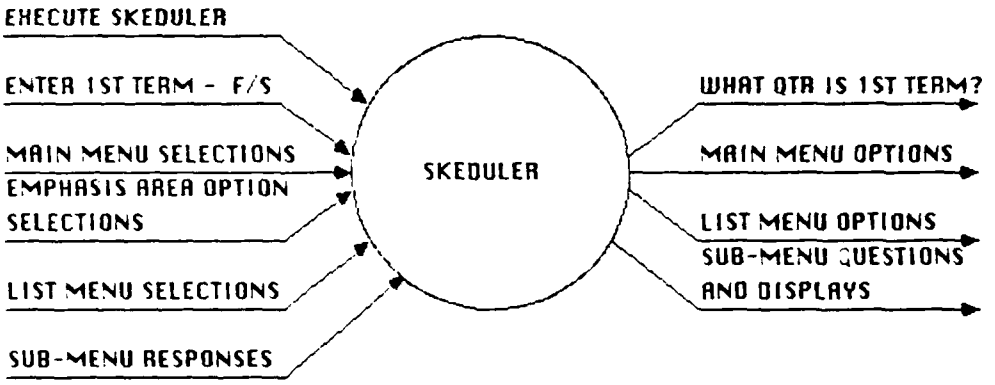
Conversions: to MAN DAYS = MM x 19
to MAN HOURS= MM x 152

(.2052 MM)(19) = 3.9 MD (1 day/term)
(.2052 MM)(152)= 31 MH (8 hrs/term)

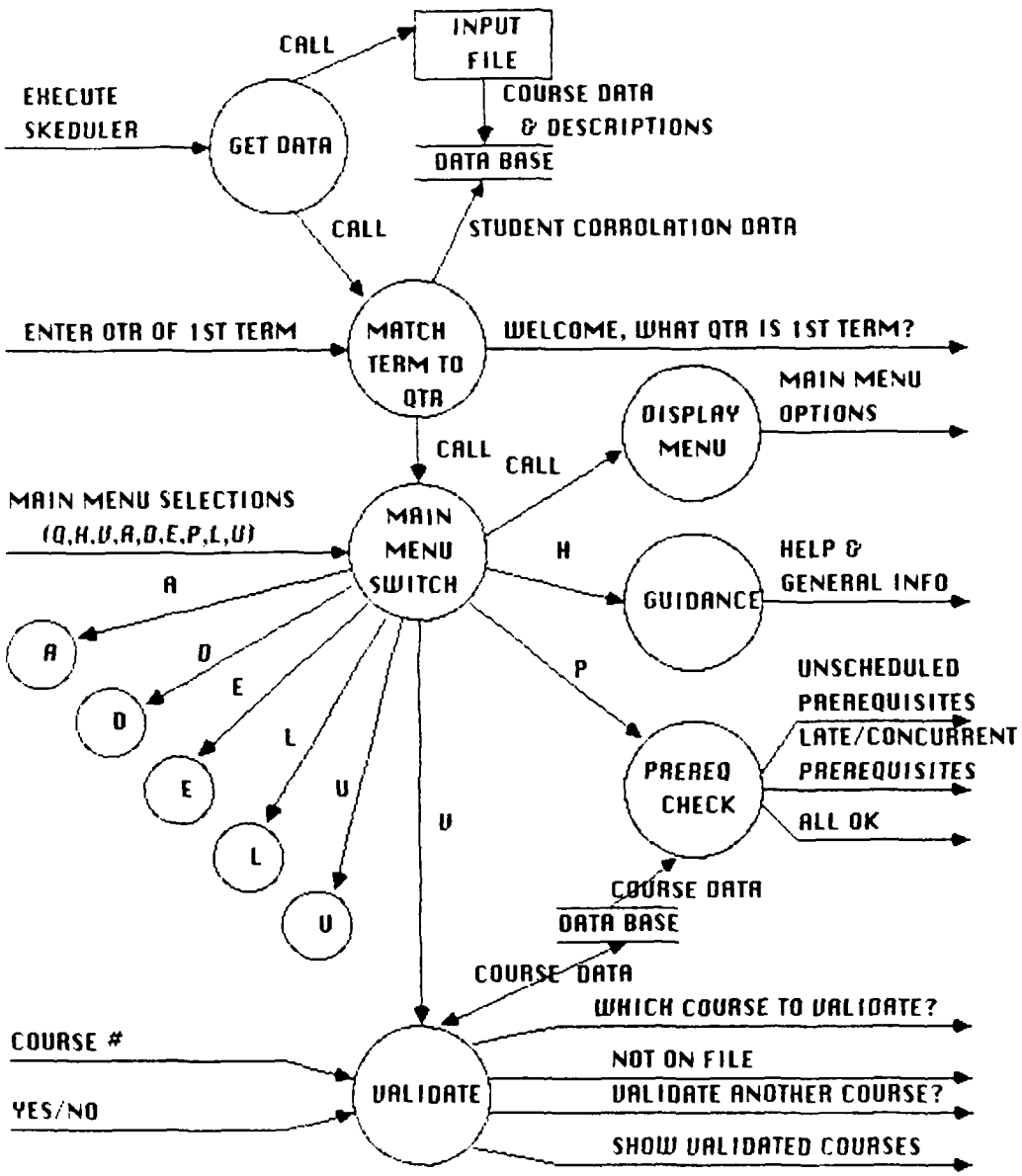
APPENDIX C

DATA FLOW DIAGRAM

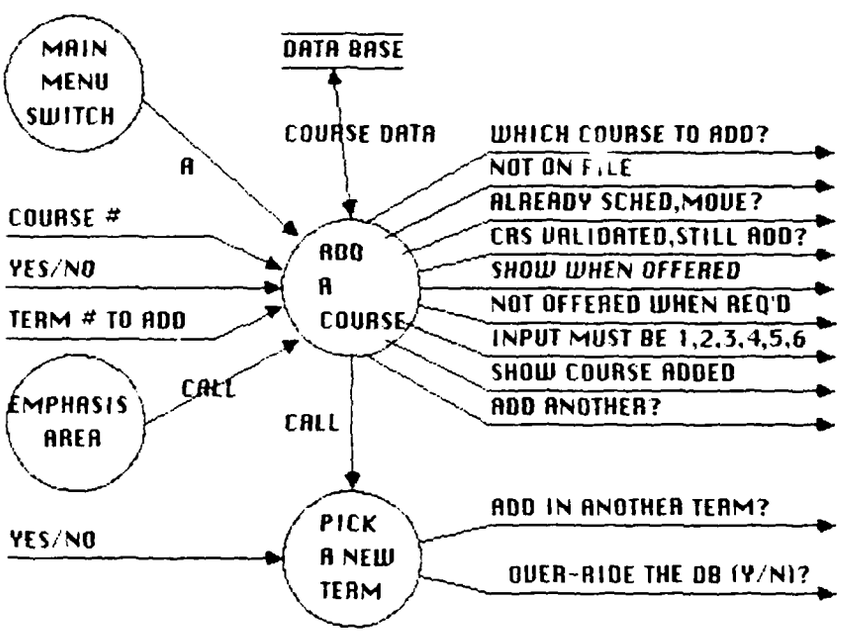
FUNDAMENTAL DATA FLOW DIAGRAM

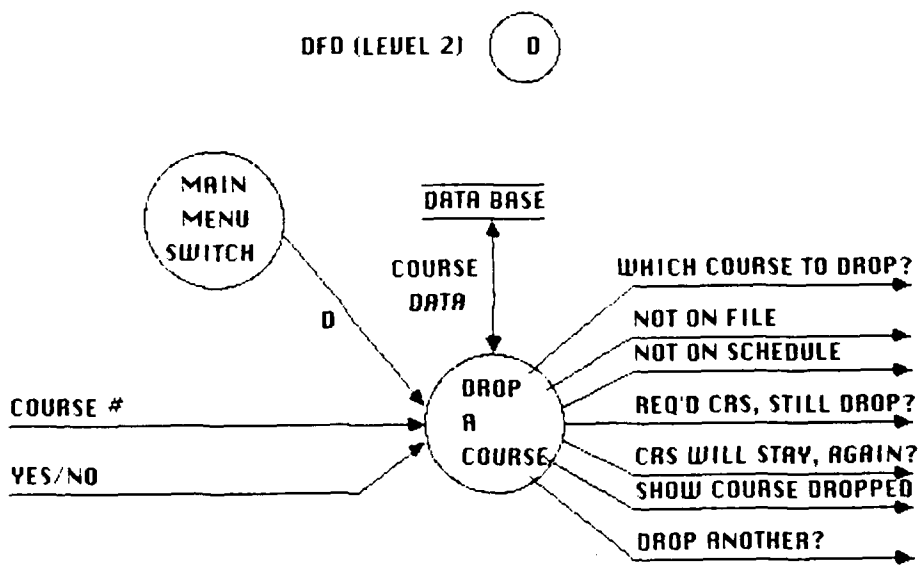


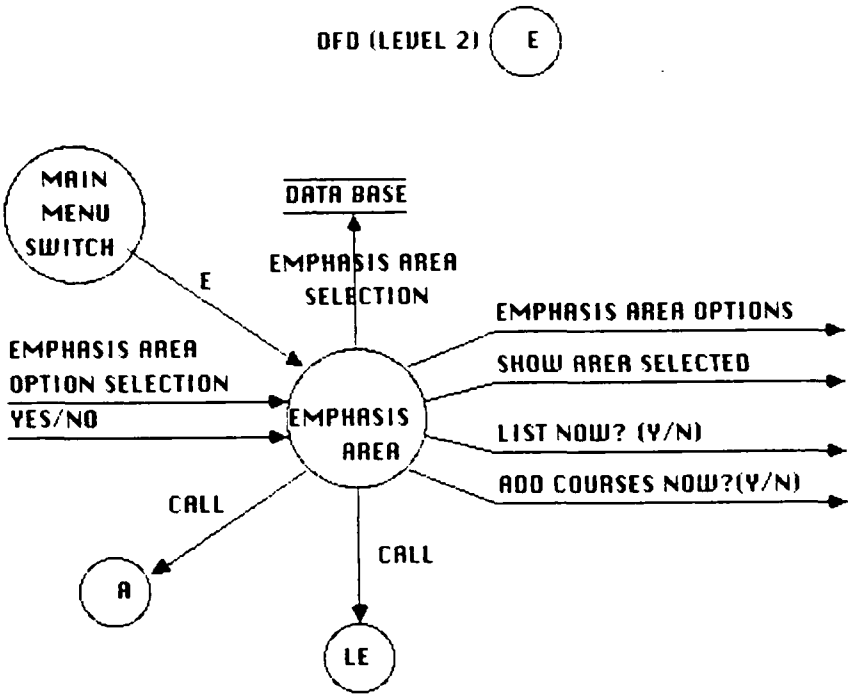
DATA FLOW DIAGRAM (LEVEL 2)

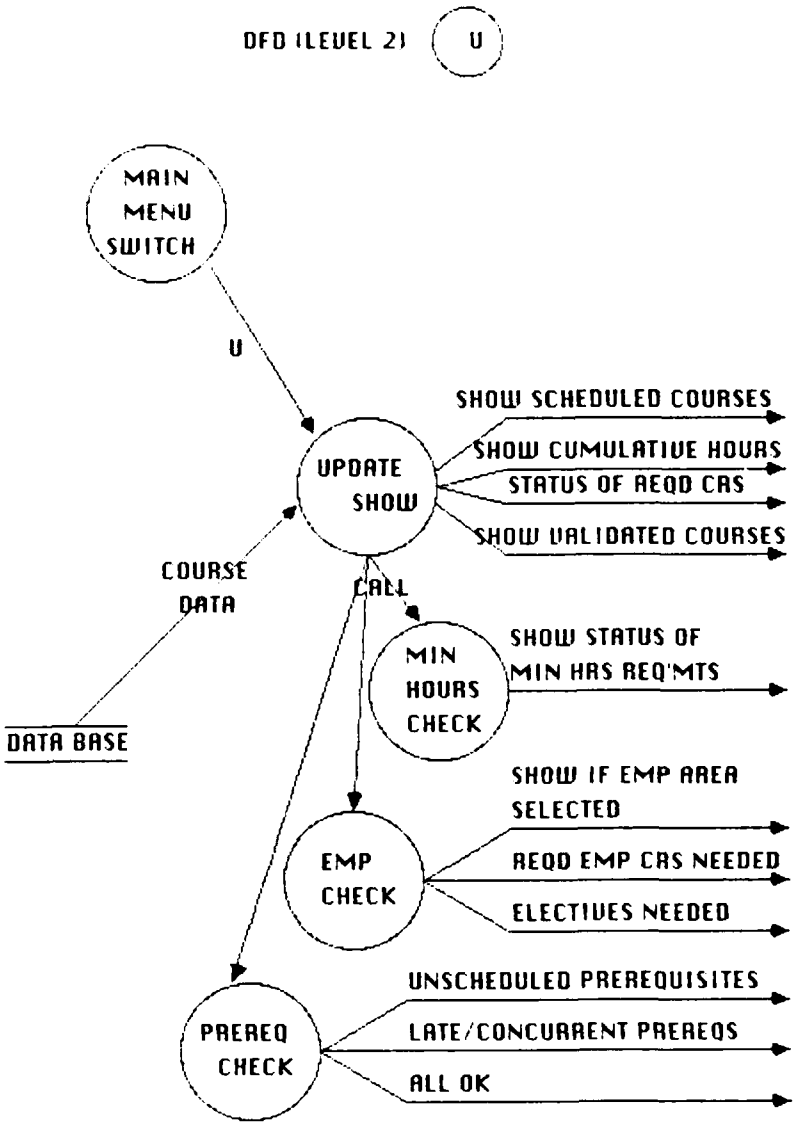


DFD (LEVEL 2) (A)

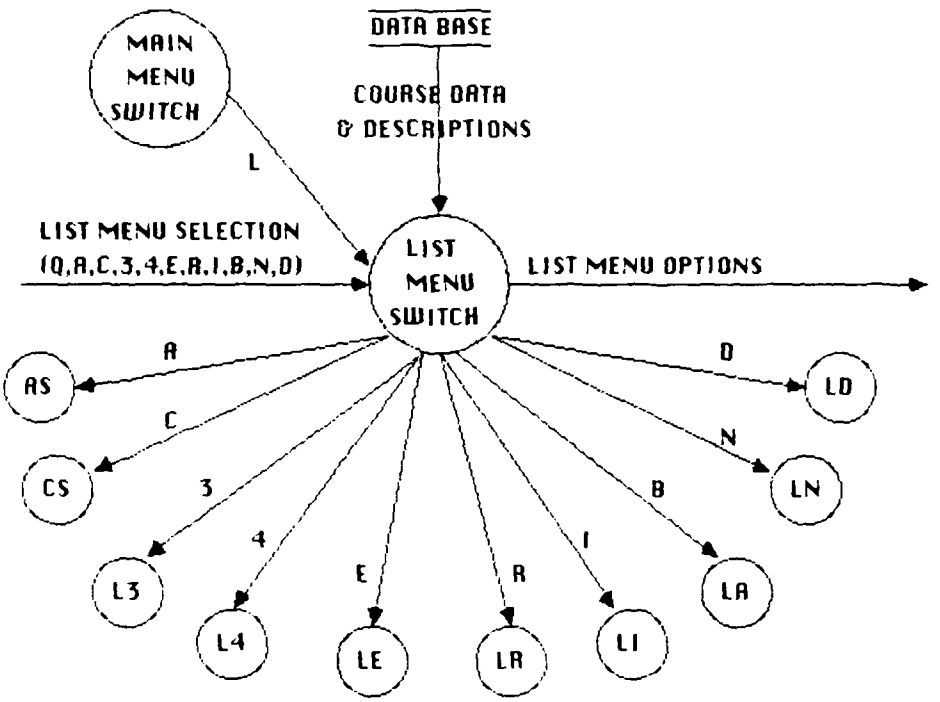

MAIN MENU SWITCH
A
DATA BASE
COURSE DATA
COURSE #
YES/NO
TERM # TO ADD
ADD A COURSE
EMPHASIS AREA
CALL
CALL
WHICH COURSE TO ADD?
NOT ON FILE
ALREADY SCHED,MOVE?
CRS VALIDATED,STILL ADD?
SHOW WHEN OFFERED
NOT OFFERED WHEN REQ'D
INPUT MUST BE 1,2,3,4,5,6
SHOW COURSE ADDED
ADD ANOTHER?
YES/NO
PICK A NEW TERM
ADD IN ANOTHER TERM?
OVER-RIDE THE DB (Y/N)?

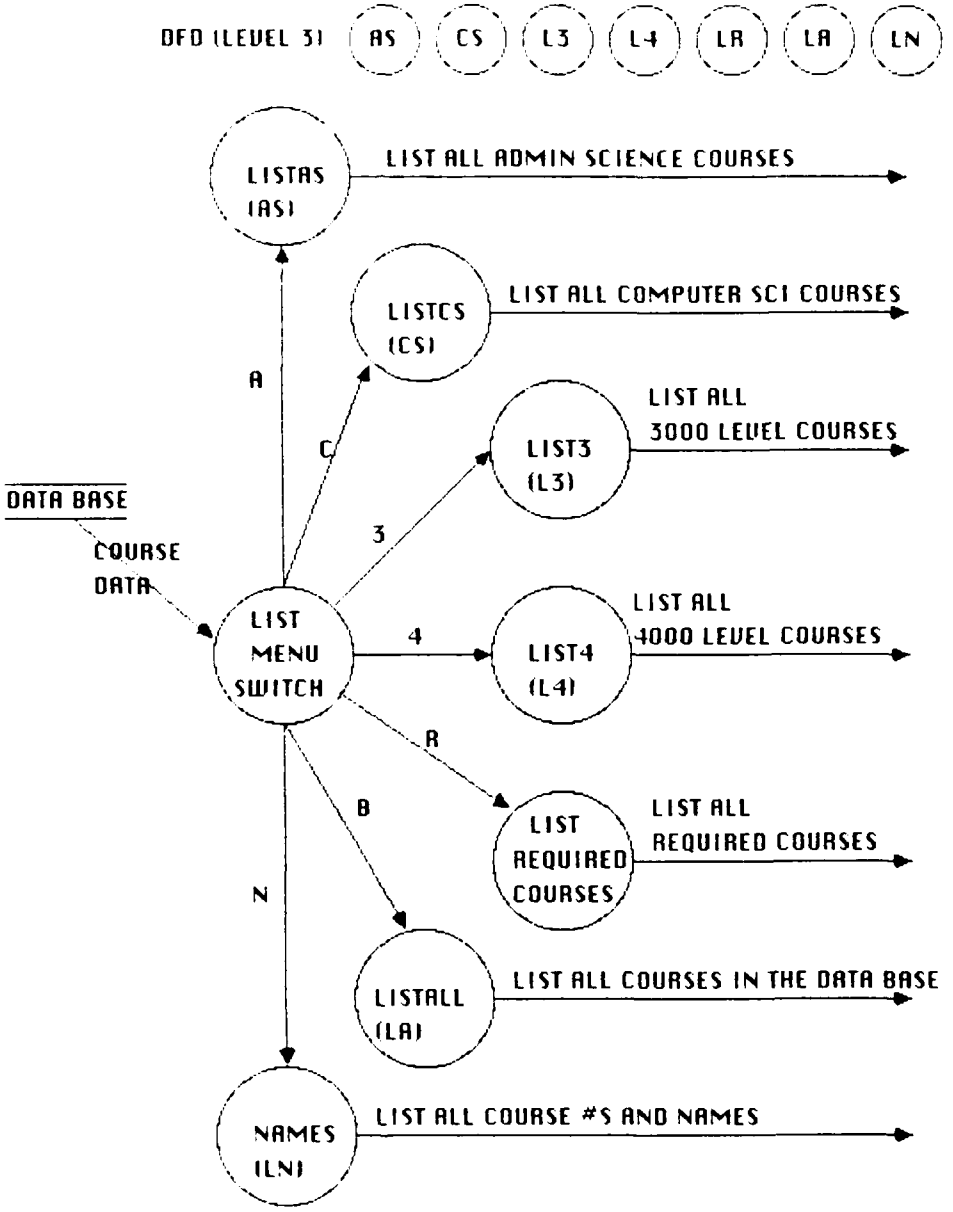

DFD (LEVEL 2)
D
MAIN
MENU
SWITCH
D
DATA BASE
COURSE
DATA
COURSE #
YES/NO
DROP
A
COURSE
WHICH COURSE TO DROP?
NOT ON FILE
NOT ON SCHEDULE
REQ'D CRS, STILL DROP?
CRS WILL STAY, AGAIN?
SHOW COURSE DROPPED
DROP ANOTHER?

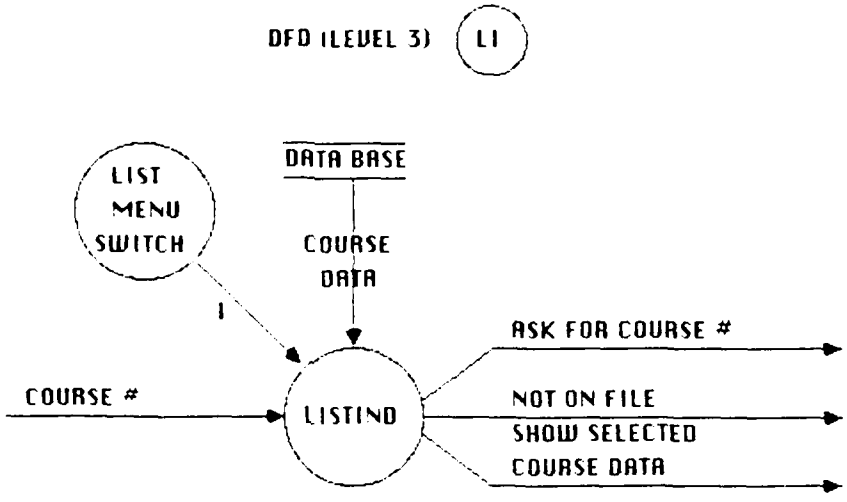

DFD (LEVEL 2)
E
MAIN MENU SWITCH
E
DATA BASE
EMPHASIS AREA SELECTION
EMPHASIS AREA OPTIONS
SHOW AREA SELECTED
EMPHASIS AREA OPTION SELECTION
YES/NO
EMPHASIS AREA
LIST NOW? (Y/N)
ADD COURSES NOW?(Y/N)
CALL
A
CALL
LE

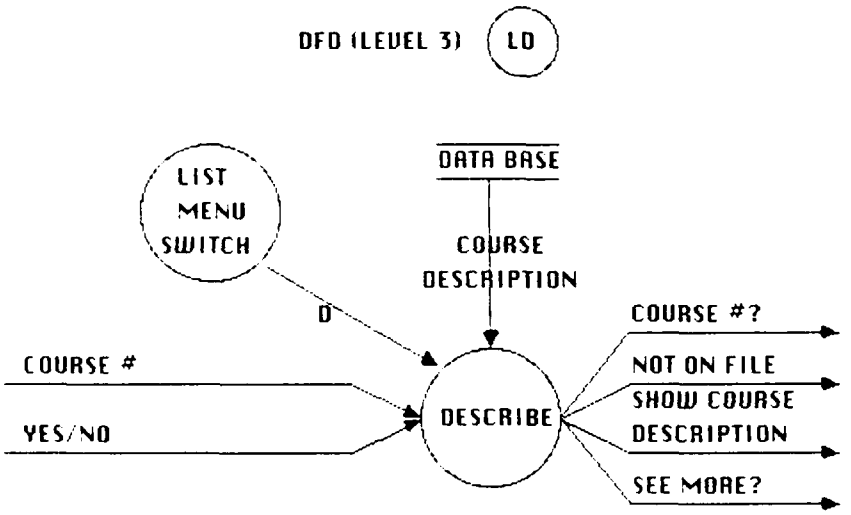

DFD (LEVEL 2)
U
MAIN MENU SWITCH
U
UPDATE SHOW
SHOW SCHEDULED COURSES
SHOW CUMULATIVE HOURS
STATUS OF REQD CRS
SHOW VALIDATED COURSES
COURSE DATA
CALL
DATA BASE
MIN HOURS CHECK
SHOW STATUS OF MIN HRS REQ'MTS
SHOW IF EMP AREA SELECTED
EMP CHECK
REQD EMP CRS NEEDED
ELECTIVES NEEDED
UNSCHEDULED PREREQUISITES
PREREQ CHECK
LATE/CONCURRENT PREREQS
ALL OK

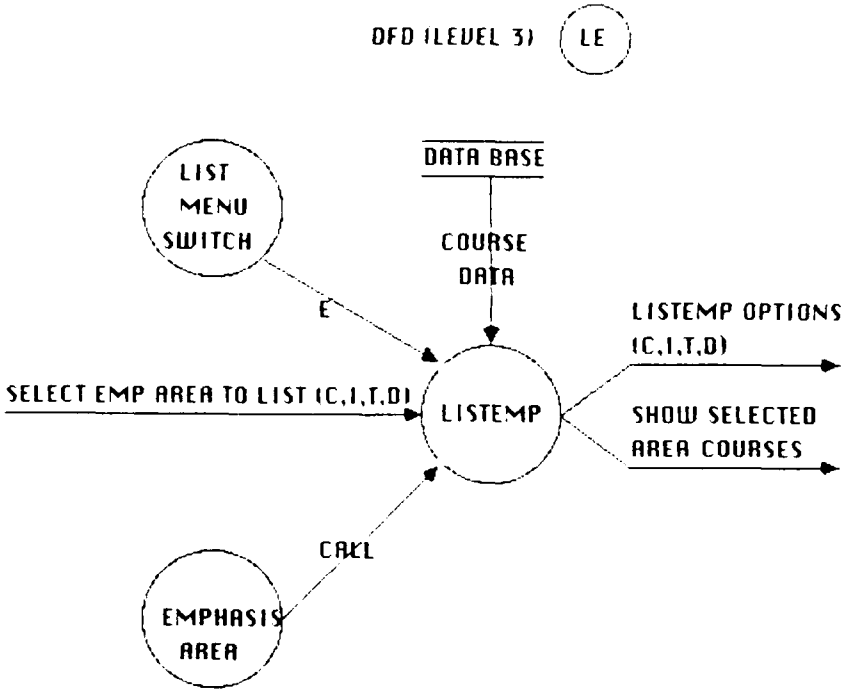
DFD (LEVEL 2) (L)


MAIN MENU SWITCH
DATA BASE
COURSE DATA & DESCRIPTIONS
L
LIST MENU SELECTION (Q,A,C,3,4,E,R,I,B,N,D)
LIST MENU SWITCH
LIST MENU OPTIONS
A
AS
C
CS
3
L3
4
L4
E
LE
R
LR
I
LI
B
LA
N
LN
D
LD


DFD (LEVEL 3)
AS
CS
L3
L4
LR
LA
LN
LISTAS (AS)
LIST ALL ADMIN SCIENCE COURSES
LISTCS (CS)
LIST ALL COMPUTER SCI COURSES
LIST3 (L3)
LIST ALL 3000 LEVEL COURSES
LIST4 (L4)
LIST ALL 4000 LEVEL COURSES
LIST REQUIRED COURSES
LIST ALL REQUIRED COURSES
LISTALL (LA)
LIST ALL COURSES IN THE DATA BASE
NAMES (LN)
LIST ALL COURSE #'S AND NAMES
DATA BASE
COURSE DATA
LIST MENU SWITCH
A
C
3
4
R
B
N


DFD (LEVEL 3)
L1
LIST
MENU
SWITCH
DATA BASE
COURSE
DATA
1
ASK FOR COURSE #
COURSE #
LISTIND
NOT ON FILE
SHOW SELECTED
COURSE DATA


DFD (LEVEL 3)
LD
LIST
MENU
SWITCH
DATA BASE
COURSE
DESCRIPTION
0
COURSE #
YES/NO
DESCRIBE
COURSE #?
NOT ON FILE
SHOW COURSE
DESCRIPTION
SEE MORE?


DFD (LEVEL 3)
LE
LIST
MENU
SWITCH
DATA BASE
COURSE
DATA
E
LISTEMP OPTIONS
(C,I,T,D)
SELECT EMP AREA TO LIST (C,I,T,D)
LISTEMP
SHOW SELECTED
AREA COURSES
CALL
EMPHASIS
AREA

APPENDIX D

DATA DICTIONARY

INDEX:

	CATAGORY		PAGES
1)	DATA SOURCES		39-40
2)	INFORMATION STORES		40
3)	TRANSACTION CENTERS		41
4)	TRANSFORMS		41-44
5)	DATA FLOWS:	INPUT	45-47
		OUTPUT	48-52
		INTERNAL	53-54

Note: dictionary entries are in alphabetical order within each catagory

****

DATA SOURCES:

1. INPUT FILE - called at the start of program execution ("LOAD DATABASE PROCEDURE" - see transforms) to load the "DATABASE" (see info stores) with the following course information:

   A) SET 1 - COURSE DATA (FIRST 57 ENTRIES)

COL(S)	CONTENTS	
1- 2	COURSE ID	
4- 7	COURSE NUMBER	
9	HOURS	
11	TERM ASSIGNED	(PRESET SCHEDULE) (= ACTUAL TERM PLUS 1) (1 = VALIDATED)
13-14	CURRICULUM	
16-17	PREREQUISITE #1 -	COURSE ID
19-23		COURSE NUMBER

24-25	PREREQUISITE #2 - COURSE ID
27-30	COURSE NUMBER
32-33	PREREQUISITE #3 - COURSE ID
35-38	COURSE NUMBER
40-41	PREREQUISITE #4 - COURSE ID
43-46	COURSE NUMBER
48	REQUIRED COURSE CODE (1 = YES / 0 = NO)
50	OFFERED IN THE: FALL (1 = YES / 0 = NO)
52	WINTER (" / ")
54	SPRING (" / ")
56	SUMMER (" / ")
	EMPHASIS AREA COURSES:
58	DSS (2 = REQD / 1 = ELECTIVE / 0 = NA)
60	TS (" / " / ")
62	ICN (" / " / ")
64	CCNO (" / " / ")

B) SET 2-COURSE DESCRIPTIONS (SECOND 57 ENTRIES)
(EACH ENTRY CONSISTS OF A 10 LINE BLOCK)

COL(S)	CONTENTS
1-73	COURSE NUMBER, NAME, LECTURE/LAB HOURS, DESCRIPTION, PREREQUISITES

****

INFORMATION STORES:

1. DATABASE - initially loaded from the "INPUT FILE" (see data sources) with course data and the preset matrix. DB is changed when the user modifies the schedule during program execution. Also contains minimum hour accounting variables, term to quarter correlators, and emphasis area selection variables.

****

****

TRANSACTION CENTERS:

1. L - see LIST MENU SWITCH

2. LIST MENU SWITCH - called from the "MAIN MENU SWITCH" to display menu options and transfer lister menu selections to appropriate lister procedures.

3. MAIN MENU SWITCH - procedure to call the "MAIN MENU DISPLAY" (see TRANSFORMS) and transfer menu option selections to appropriate procedures

****

TRANSFORMS:

1. A - see ADD_A_COURSE

2. ADD_A_COURSE - procedure called from the "MAIN MENU SWITCH"(see TRANSACTION CENTERS). allows the user to add a course to his schedule

3. AS - see LISTAS

4. CS - see LISTCS

5. D - see DROP_A_COURSE

6. DESCRIBE - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list selected course descriptions

7. DISPLAY_MENU - called from "MAIN MENU SWITCH" (see TRANSACTION CENTERS) to display the menu on the terminal screen. the display will remain on the screen until a menu selection is made

8. DROP_A_COURSE - procedure called from the "MAIN MENU SWITCH"(see TRANSACTION CENTERS). allows the user to remove a course from his schedule

9. E - see EMPHASIS_AREA

10. EMP CHECK - called from "UPDATE/SHOW" to see if an emphasis area has been selected and if so, to tell the user if the required emphasis course or any emphasis electives need to be scheduled

11. GET_DATA - procedure to draw data from the "INPUT FILE" (see DATA SOURCES) and place it in the "DATABASE" (see INFORMATION STORES)

12. GUIDANCE - called from "MAIN MENU SWITCH" (see TRANSACTION CENTERS) to display help data and general information about the program

13. L3 - see LIST3

14. L4 - see LIST4

15. LA - see LISTALL

16. LD - see DESCRIBE

17. LE - see LISTEMP

18. LI - see LISTIND

19. LIST3 - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list all 3000 level courses in the "DATABASE"

20. LIST4 - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list all 4000 level courses in the "DATABASE"

21. LISTALL - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list all courses in the "DATABASE"

22. LISTAS - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list all Admin Science courses in the "DATABASE"

23. LISTCS - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list all Computer Science courses in the "DATABASE"

24. LISTEMP - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list all emphasis area courses in the "DATABASE"

25. LISTIND - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list individual courses in the "DATABASE"

26. LIST_REQUIRED_COURSES - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list all required courses in the "DATABASE"

27. LN - see NAMES

28. LR - see LIST_REQUIRED_COURSES

29. MATCH_TERM_TO_QTR - procedure to determine when the student's first term is so that term numbers can be corrolated to academic quarters (e.g. term 1 = fall, term 2 = winter, etc.)

30. MIN HRS CHECK - procedure called from "UPDATE/SHOW" to check if there are deficiencies in scheduling AS,CS, graduate level and 4000 level courses. Also checks to see if enough courses have been scheduled per term. Informs the user of any deficiencies.

31. NAMES - procedure called from "LISTER SWITCH" (see TRANSACTION CENTERS) to list selected course descriptions

32. PICK_A_NEW_TERM - called from "ADD" when a user tries to add a course in a term that it is not offered in (according to the "DATABASE"). allows the user to override the "DATABASE".

33. PREREQ CHECK - called from the "MAIN MENU SWITCH" (see TRANSACTION CENTERS) to check all scheduled course prerequisites and give the status to the user.

34. EMPHASIS_AREA - called from the "MAIN MENU SWITCH" (see TRANSACTION CENTERS) to allow the student select one of four offered emphasis areas. this keys the program as to which courses to check in obtaining emphasis area requirements fulfillment.

35. SKEDULER - interactive, terminal oriented, menu driven program to aid CSM students in course selection and scheduling.

36. U - see Update_Show

37. UPDATE/SHOW - called from the "MAIN MENU SWITCH" (see TRANSACTION CENTERS) to update the users schedule and give him the status of minimum requirements fulfillment. this module calls other modules to help accomplish this task (i.e. MIN HRS CHECK, EMP CHECK, PREREQ CHECK)

38. VALIDATE - called from the "MAIN MENU SWITCH" (see TRANSACTION CENTERS). allows the user to indicate which courses have been validated. This allows those those courses to count as prerequisites while not appearing on the schedule. at the end of validation, gives the user a summary of validated courses.

****

****

DATA FLOWS (INPUT):

1. COURSE # - a six character (2 alpha, 4 numeric) input string from the keyboard to list a course # in response to questions from the "VALIDATE","ADD", "DROP", "LISTIND", and "DESCRIBE" transforms.

2. EMPHASIS AREA OPTION SELECTIONS -
   - "C" - character input from the keyboard to select emphasis area Computer Center and Network Operations in response to "EMP AREA OPTION MENU" (see DATA FLOWS (output))
   - "I" - character input from the keyboard to select emphasis area Information and Computer Networks in response to "EMP AREA OPTION MENU" (see DATA FLOWS (output))
   - "T" - character input from the keyboard to select emphasis area Tactical Systems in response to "EMP AREA OPTION MENU" (see DATA FLOWS(output))
   - "D" - character input from the keyboard to select emphasis area Decision Support Systems in response to "EMP AREA OPTION MENU" (see DATA FLOWS (output))

3. ENTER QTR OF FIRST TERM - character input from the keyboard in response to the question "which academic quarter is your first term in?" (see DATA FLOWS (output)).

4. EXECUTE SKEDULER - "FW SKEDULER" entered to execute the program

5. LIST MENU SELECTIONS - responses to List Menu Options
   - "Q" - character input from the keyboard to stop listing and return to the main menu
   - "A" - character input from the keyboard to see all

Admin Science courses. activates "LISTAS" (see TRANSFORMS)

"C" - character input from the keyboard to see all Computer Science courses. activates "LISTCS" (see TRANSFORMS)

"3" - character input from the keyboard to see all 3000 level courses. activates "LIST3" (see TRANSFORMS)

"4" - character input from the keyboard to see all 4000 level courses. activates "LIST4" (see TRANSFORMS)

"E" - character input from the keyboard to see all Emphasis Area courses. activates "LISTEMP" (see TRANSFORMS) (see also LIST EMPHASIS AREA OPTION SELECTIONS above)

"R" - character input from the keyboard to see all required courses. activates "LISTREQ" (see TRANSFORMS)

"I" - character input from the keyboard to see info on an individual course. activates "LISTIND" (see TRANSFORMS)

"N" - character input from the keyboard to see all course names. activates "NAMES" (see TRANSFORMS)

"D" - character input from the keyboard to see description of an individual course. activates "DESCRIBE" (see TRANSFORMS)

6. MAIN MENU SELECTIONS - keyboard responses to Main Menu Options to select routing to an appropriate TRANSFORM

"H" - character input to select "GUIDANCE"

"V" - character input to select "VALIDATE"

"A" - character input to select "ADD"

"D" - character input to select "DROP"

"E" - character input to select "SELECT EMP AREA"
"L" - character input to select "LISTER SWITCH"
"U" - character input to select "UPDATE/SHOW"
"P" - character input to select "PREREQ CHECK"
"Q" - character input to terminate execution

7. SELECT EMP AREA TO LIST - keyboard responses to "LIST EMP AREA MENU" (see DATA FLOWS (output))
   "C" - character input to see emphasis area Computer Center and Network Operations
   "I" - character input to see emphasis area Information and Computer Networks
   "T" - character input to see emphasis area Tactical Systems
   "D" - character input to see emphasis area Decision Support Systems

8. SUB-MENU RESPONSES - all other input data flows not covered in the other fundamental model inputs

9. TERM # TO ADD - character input (1,2,3,4,5, or 6) from the keyboard in response to the following question from "ADD" TRANSFORM:
   "what term do you want to add this course in?"

10. YES/NO - character input from the keyboard in response to any yes/no question

****

****

DATA FLOWS (OUTPUT):

1. ADD ANOTHER? - terminal screen output asking the user if he wants to add another course. requires a yes/no response.

2. ADD COURSES NOW? - ask the user if he wants to add any of the emphasis area courses now

3. ADD IN ANOTHER TERM? - ask the user if he wishes to change his mind regarding his term selection.

4. ALL OK - tell the user that all course prerequisites have been scheduled properly

5. ALREADY SCHEDULED, MOVE? - tell the user he has asked to add a course that is already on the schedule and that if he adds it, the program will move the course to the location of his latest choice.

6. ASK FOR COURSE # TO SEE - terminal screen output resulting from LIST MENU OPTION SELECTION "I". (see DATA FLOWS (input))

7. COURSE # - terminal screen output resulting from LIST MENU OPTION SELECTION "D" asking for the # of the course description to see.

8. CRS VALIDATED, STILL ADD? - tell the user he has asked to add a course that he previously validated. ask if that is his intention.

9. CRS WILL STAY, AGAIN? - tell the user he has cancelled his drop request and ask if he wishes to drop another course.

10. DROP ANOTHER? - terminal screen output asking the user if he wants to drop another course. (y/n response)

11. ELECTIVES NEEDED - tell if enough emphasis area electives have been scheduled

12. EMP AREA OPTIONS - terminal screen output menu that requires a single character response (C/I/T/D). related to "SEL EMP AREA" TRANSFORM.

13. HELP & GENERAL INFO - terminal scrren output in response to user selection of Main Menu Option "H". tells the user about the program

14. INPUT MUST BE 1,2,3,4,5,6 - tell the user his term selection must be numeric between 1 and 6.

15. LATE/CONCURRENT PREREQUISITES - tell the user which prerequisites are not scheduled or are scheduled concurrently with the master course

16. LIST ALL 3000 LEVEL COURSES - terminal screen output resulting from LIST MENU OPTION selection "3".

17. LIST ALL 4000 LEVEL COURSES - terminal screen output resulting from LIST MENU OPTION selection "4".

18. LIST ALL ADMIN SCIENCE COURSES - terminal screen output resulting from LIST MENU OPTION selection "A".

19. LIST ALL COMPUTER SCIENCE COURSES - terminal screen output resulting from LIST MENU OPTION selection "C".

20. LIST ALL COURSE #s AND NAMES - terminal screen output resulting from LIST MENU OPTION selection "N".

21. LIST ALL COURSES IN DATABASE - terminal screen output resulting from LIST MENU OPTION selection "B".

22. LIST ALL REQUIRED COURSES - terminal screen output resulting from LIST MENU OPTION selection "R".

23. LIST MENU OPTIONS - terminal screen output to show the user listing options. (single character response)

24. LISTEMP OPTIONS - terminal screen output resulting from LIST MENU OPTION selction "E". displays a menu of emp area options to look at. (C/I/T/D)

25. MAIN MENU OPTIONS - terminal screen output to show the user his options in operating the program. requires a single character response.

26. LIST NOW? - ask the user if he wants to see the courses available in his selected emphasis area

27. NOT OFFERED WHEN REQ'D - tell the user he has asked to add the course in a term that it is not offered (according to the data base).

28. NOT ON FILE - terminal screen output response that appears whenever the user enters a course # for validation, adding, dropping, or listing that is not resident in the "DATABASE"

29. NOT ON SCHEDULE - tell the user that the course he has asked to drop is not on the current schedule

30. OVER-RIDE THE DATA BASE? (Y/N) - tell the user he can over-ride the data base and add a course in a term not listed as valid

31. REQD CRS, STILL DROP? - tell the user he is about to drop a required course and ask him if he still wants to do it.

32. REQD EMP CRS NEEDED - tell if the emphasis area required course has been scheduled

33. SEE MORE? - terminal screen output asking the user if he wants to get a description of another course.

34. SHOW AREA SELECTED - show the user the title of the emphasis area he has selected

35. SHOW COURSE DESCRIPTION - terminal screen output resulting from course # input within LISTER module

36. SHOW COURSES ADDED/DROPPED - terminal screen output showing which courses have been added or dropped from the most recent request

37. SHOW CUMULATIVE HOURS - lists total hours scheduled in the following catagories:
    1. AS hours
    2. CS hours
    3. grad level hours
    4. 4000 level hours
    5. total hours

38. SHOW IF EMP AREA SELECTED - tell if emphasis area has not been selected.

39. SHOW SCHEDULED COURSES - activated from "Update_Show" TRANSFORM. Shows courses scheduled by term.

40. SHOW SELECTED AREA COURSES - terminal screen output appearing in response to EMPHASIS LIST MENU option selctions:
    "C" - courses for Computer Center and Network Operations
    "I" - courses for Information and Computer Networks
    "T" - courses for Tactical Systems
    "D" - courses for Decision Support Systems

41. SHOW SELECTED COURSE DATA - terminal screen output resulting from course # input in selecting an individual course to list.

42. SHOW STATUS OF MIN HRS REQ'MTS - tells if enough hours have been scheduled in AS, CS, grad level, 4000 level and if enough courses have been scheduled per term to meet minimum requirements

43. SHOW VALIDATED COURSES - terminal screen output that lists all validated courses. appears after a validation session and during schedule update.

44. SHOW WHEN OFFERED - show the user (by term and academic qtr) when his selected course is offered.

45. STATUS OF REQD CRS - Tell if all required courses are scheduled or validated.

46. SUB-MENU QUESTIONS AND DISPLAYS - all output data flows not covered in the other fundamental model outputs.

47. UNSCHEDULED PREREQUISITES - tell the user when missing prerequisites must be scheduled (by term)

48. VALIDATE ANOTHER COURSE? - terminal screen output asking the user if he wants to validate another course.(y/n)

49. WELCOME, WHAT QTR IS 1ST TERM IN? - terminal screen output asking for data to use in corrolating student term # to academic quarter.

50. WHAT QTR IS FIRST TERM IN - terminal screen output seen upon program execution. the response will be used to corrolate student terms (1-6) to academic quarters.

51. WHICH COURSE TO ADD/DROP? - terminal screen output asking the user which course to add or drop. requires a 6 character (2 alpha, 4 numeric) response.

52. WHICH COURSE TO VALIDATE - terminal screen output that appears when "VALIDATE" TRANSFORM is entered. requires a 6 character (2 alpha, 4 numeric) response.

****

****

DATA FLOWS (INTERNAL):

1. 3 -(from LIST MENU SWITCH)- control data to call LIST3
2. 4 -(from LIST MENU SWITCH)- control data to call LIST4
3. A -(from LIST MENU SWITCH)- control data to call LISTAS
4. A -(from MAIN MENU SWITCH)- control data to call ADD_A_COURSE
5. B -(from LIST MENU SWITCH)- control data to call LISTALL
6. C -(from LIST MENU SWITCH)- control data to call LISTCS
7. CALL - internal control linkage from superordinate modules to call subordinate modules based on sequence or input from the keyboard
8. COURSE DATA (AND DESCRIPTIONS) - see "INPUT FILE" DATA SOURCE
9. D -(from LIST MENU SWITCH)- control data to call DESCRIBE
10. D -(from MAIN MENU SWITCH)- control data to call DROP_A_COURSE
11. E -(from LIST MENU SWITCH)- control data to call LISTEMP
12. E -(from MAIN MENU SWITCH)- control data to call EMPHASIS_AREA
13. EMPHASIS AREA SELECTIONS - this code is stored in the "DATABASE" to show which emphasis area has been selected. used in determining if the required or elective emphasis area courses have been scheduled
14. H -(from MAIN MENU SWITCH)- control data to call GUIDANCE

15. I -(from LIST MENU SWITCH)- control data to call LISTIND

16. L -(from MAIN MENU SWITCH)- control data to call LISTER

17. N -(from LIST MENU SWITCH)- control data to call NAMES

18. P -(from MAIN MENU SWITCH)- control data to call PREREQUISITE_CHECK

19. R -(from LIST MENU SWITCH)- control data to call LIST_REQUIRED_CRS

20. STUDENT CORROLATION DATA - academic quarters are assigned to term #s and loaded to the "DATABASE" for later use in course selection. Two options are available:

TERM	OPTION 1	OPTION 2
1	fall	spring
2	winter	summer
3	spring	fall
4	summer	winter
5	fall	spring
6	winter	summer

21. U -(from MAIN MENU SWITCH)- control data to call UPDATE_SHOW

22. V -(from MAIN MENU SWITCH)- control data to call VALIDATE

****

APPENDIX E

USER'S GUIDE

TABLE OF CONTENTS

SUBJECT	PAGE
1. INTRODUCTION	56
2. REQUIREMENTS	56
3. PRIOR TO EXECUTION	57
4. EXECUTION	58
5. MAIN MENU	59
5.1 H - HELP AND GENERAL INFORMATION	60
5.2 V - VALIDATE A COURSE	62
5.3 A - ADD A COURSE (TO THE SCHEDULE)	63
5.4 D - DROP A COURSE (FROM THE SCHEDULE)	66
5.5 E - SELECT AN EMPHASIS AREA	68
5.6 L - LIST MENU	70
5.6.1 A - LIST ADMIN SCIENCE COURSES	71
5.6.2 C - LIST COMPUTER SCIENCE COURSES	71
5.6.3 3 - LIST 3000 LEVEL COURSES	71
5.6.4 4 - LIST 4000 LEVEL COURSES	71
5.6.5 R - LIST REQUIRED COURSES	71
5.6.6 B - LIST ALL COURSES IN THE DATABASE	71
5.6.7 E - LIST EMPHASIS AREA COURSES	72
5.6.8 I - LIST DATA ON ANY INDIVIDUAL COURSE	73
5.6.9 N - LIST ALL COURSE NAMES	74
5.6.10 D - LIST COURSE DESCRIPTIONS	74
5.6.11 Q - QUIT LISTING/RETURN TO MAIN MENU	75
5.7 U - UPDATE YOUR SCHEDULE AND SEE STATUS OF REQUIREMENTS	76
5.8 P - SEE IF ALL PREREQUISITES ARE FULFILLED	79
5.9 Q - QUIT "SKEDULER" (TERMINATE EXECUTION)	79
6. OBTAINING A HARD COPY OF YOUR SCHEDULE	80

****

1.

INTRODUCTION

This program is an interactive, terminal oriented, menu driven software package designed to aid CSM students in course selection and scheduling. The program provides each student with a preset schedule that contains all required courses. The student must build on that schedule when selecting emphasis area courses. The program will keep track of all prerequisites, validated courses, graduation requirements, tentative offering times and emphasis area requirements. The program will prevent the student from making scheduling errors by continuously conducting requirements checks and alerting the student to actual or potential problems. Upon completion of each SKEDULER session, the program will provide the student with a complete academic schedule that can be saved and transmitted either electronically or by hard copy.

****

2.

REQUIREMENTS

This program will be located on each students A-disk for his/her individual use.

****

****

3.

PRIOR TO EXECUTION

Due to the size of the program, storage must be increase prior to execution. While in CMS, enter:

```
DEFINE STORAGE 1M
```

You will see the following after entered:

```
define storage 1m
STORAGE = 01024 K
CP ENTERED; DISABLED WAIT PSW '00020000 00000000'
```

Now you must reenter CMS by typing the following:

```
I CMS
```

After entry you will see the following:

```
i cms
CMS 2.1 - 10/19/84
```

Now you are ready to execute the program.

****

****

4.

TO EXECUTE THE PROGRAM

To execute "SKEDULER" simply enter the following:

```
PW SKEDULER
```

You will see the following:

Execution begins...

```
**************************************
*   ****************************   *
*   *   WELCOME TO "SKEDULER"    *   *
*   ****************************   *
**************************************
```

WHAT QTR OF THE YEAR IS YOUR FIRST TERM IN?

```
ENTER "F" FOR FALL    (OCT-DEC)
ENTER "S" FOR SPRING  (APR-JUN)
```

Enter "f" or "s" here. Any other entry will cause the question to reappear. A correct entry will lead you to the Main Menu.

****

****

5.

MAIN MENU

The main menu will appear as shown below after responding to the opening question and following completion of any of the main menu functions.

```
******************** MAIN MENU ************************
*                                                     *
*   ENTER "H" FOR HELP AND GENERAL INFORMATION        *
*   ENTER "V" TO VALIDATE A COURSE                    *
*   ENTER "A" TO ADD  A COURSE TO   THE SCHEDULE      *
*   ENTER "D" TO DROP A COURSE FROM THE SCHEDULE      *
*   ENTER "E" TO SELECT AN EMPHASIS AREA              *
*   ENTER "L" TO LIST COURSES AVAILABLE               *
*   ENTER "U" TO SEE YOUR UPDATED SCHEDULE AND        *
*             TO SEE YOUR STATUS ON FULFILLMENT       *
*             OF MINIMUM REQUIREMENTS                 *
*   ENTER "P" TO SEE IF PREREQUISITES HAVE BEEN MET   *
*   ENTER "Q" TO QUIT                                 *
*                                                     *
******************** MAIN MENU ************************
```

5.1

MAIN MENU SELECTION H

ENTER "H" FOR HELP AND GENERAL INFORMATION
An "h" will cause the following information to appear:

Debug? - If this should appear in the top left corner of your screen , you have made an input error that has terminated the program.
Enter "q" (without quotes) to exit the debug routine and re-enter CMS.

MORE...- If this should appear in the lower right corner of your screen, you must depress the "ALT" key and the "PA2" key (or the "CLEAR" key) simultaneously to see the next screen of information.
NEVER try to enter data when this signal is on your screen: it WILL terminate the program.

THIS PROGRAM IS DESIGNED TO HELP COMPUTER SYSTEM MGMT STUDENTS (CURRIC #367) PREPARE THEIR ACADEMIC SCHEDULES. EACH STUDENT IS PROVIDED WITH A PRESET SCHEDULE THAT CONTAINS ALL REQUIRED COURSES. THE STUDENT MUST BUILD ON THAT SCHEDULE IN ORDER TO MEET MINIMUM GRADUATION REQUIREMENTS:

CATEGORY	MINIMUM
ADMIN SCIENCE HOURS	24
COMPUTER SCIENCE HOURS	16
GRADUATE LEVEL HOURS	48
4000 LEVEL HOURS	12 (OF 48 GRAD)
COURSES PER TERM (1-3)	4
" " " (4-6)	3

PLUS:

1. SCHEDULE OR VALIDATE ALL REQUIRED COURSES
2. SELECT AN EMPHASIS AREA WITH:
    A. ONE REQUIRED COURSE
    B. THREE ELECTIVES

THIS MUST ALL BE DONE KEEPING IN MIND THAT ALL COURSES ARE NOT OFFERED EVERY QUARTER. FURTHERMORE, COURSE PREREQUISITES MUST BE SCHEDULED OR VALIDATED.

THIS PROGRAM WILL KEEP TRACK OF ALL THESE REQUIREMENTS FOR YOU AND WILL ALLOW YOU TO CREATE ANY SCHEDULE YOU DESIRE (WITHIN DATABASE LIMITATIONS).

TO OBTAIN A HARD COPY OF SCHEDULE CREATED:
A COPY OF THE LATEST ENTIRE SESSION ON "SKEDULER" WILL BE RESIDENT ON YOUR A-DISK UNDER THE FILE "SKEDULER LISTING". THAT FILE CAN BE EDITED IN XEDIT AND PRINTED ON THE VM PRINTER. IF YOUR LAST SESSION WAS OF CONSIDERABLE LENGTH, YOU MAY HAVE TO "DEFINE STORAGE 1M" PRIOR TO GOING INTO XEDIT OR YOU WILL BE TOLD THAT YOUR LISTING FILE IS TOO LARGE.

At the end of these displays, entering "ALT PA2" will return you to the main menu.

5.2

MAIN MENU SELECTION V

ENTER "V" TO VALIDATE A COURSE

A "v" will cause the following statement to appear:

ENTER THE SIX CHARACTER CODE OF THE COURSE TO VALIDATE

This refers to the two-letter course code and the four-digit course number (e.g. IS2000, MN4105, etc.). An incorrect entry or an entry for a course not resident in the database will bring about the following response:

XXXXXX IS NOT LOCATED IN THE DATABASE.
CHECK FOR INPUT ERROR

DO YOU WISH TO VALIDATE ANY OTHER COURSES?
ENTER Y FOR YES / ANY OTHER CHARACTER FOR NO

A proper entry will simply show the following:

DO YOU WISH TO VALIDATE ANY OTHER COURSES?
ENTER Y FOR YES / ANY OTHER CHARACTER FOR NO

A "yes" response will generate a request for more input
A "no" response will show the following:

VALIDATED COURSES **********

IS2000(example entry)

At this point, "ALT PA2" will take you back to the Main Menu.

5.3

MAIN MENU SELECTION A

ENTER "A" TO ADD A COURSE TO THE SCHEDULE
An "a" will cause the following statement to appear:

ENTER THE SIX CHARACTER CODE OF THE COURSE TO ADD

This refers to the two-letter course code and the four-digit course number (e.g. IS2000, MN4105, etc.). An incorrect entry or an entry for a course not resident in the database will bring about the following response:

XXXXXX IS NOT LOCATED IN THE DATABASE
DO YOU WANT TO ADD ANOTHER COURSE?
ENTER "Y" FOR YES, ANY OTHER FOR NO

A proper response for a course that is already on the current schedule will show the following:

XXXXXX IS ALREADY SCHEDULED IN TERM X
ADDING THIS COURSE WILL REMOVE IT FROM
THE CURRENT SCHEDULE LOCATION.
DO YOU STILL WANT TO ADD IT? (Y/N)

A proper entry for an unscheduled course or a "yes" response to the above question will show the following (as an example):

XXXXXX IS OFFERED IN THE /WINTER (2 & 6)//SPRING (3)/
WHAT TERM (#) DO YOU WISH TO SCHEDULE XXXXXX IN?

This question requires a response of 1,2,3,4,5 or 6. Any other response will bring about the following statement and question:

SORRY, INPUT MUST BE 1,2,3,4,5 OR 6

WHAT TERM (#) DO YOU WISH TO SCHEDULE XXXXXX IN?

The example above showed the course offered in terms 2,3 and 6. A correct numerical response (1-6) that is within the set of numbers offered by a specific course will generate the following:

XXXXXX HAS BEEN ADDED TO YOUR SCHEDULE

DO YOU WANT TO ADD ANOTHER COURSE?
ENTER "Y" FOR YES, ANY OTHER FOR NO

A correct numerical response (1-6) that is not within the set of numbers offered by a specific course will generate the following:

SORRY, XXXXXX IS NOT OFFERED IN THE (QUARTER)

DO YOU WISH TO SCHEDULE IT IN ANOTHER TERM?
ENTER "Y" FOR YES, ANY OTHER CHARACTER FOR NO

A "yes" response will generate the following question:

WHAT TERM (#) DO YOU WISH TO SCHEDULE XXXXXX IN?

A "no" response will generate the following:

IF YOU HAVE LEARNED THAT THE DATABASE IS NOT CORRECT AND THAT THIS COURSE WILL BE OFFERED AS PER YOUR REQUEST, YOU MAY SCHEDULE IT ACCORDINGLY.

IF THIS IS THE CASE, AND YOU WANT TO SCHEDULE THE COURSE AS ORIGINALLY REQUESTED,ENTER "Y". ENTER ANY OTHER CHARACTER FOR NO.

A "yes" response will add the course to your schedule.
A "no" response will generate the question of continuation:

DO YOU WANT TO ADD ANOTHER COURSE?
ENTER "Y" FOR YES, ANY OTHER FOR NO

A "yes" response will generate the question "which course to add"?
A "no" response will take you back to the Main Menu.

5.4

MAIN MENU SELECTION D

ENTER "D" TO DROP A COURSE FROM THE SCHEDULE
A "d" will cause the following statement to appear:

ENTER THE SIX CHARACTER CODE OF THE COURSE TO DROP

This refers to the two-letter course code and the four-digit course number (e.g. IS2000, MN4105, etc.). An incorrect entry or an entry for a course not resident in the database will bring about the following response:

XXXXXX IS NOT LOCATED IN THE DATABASE

DO YOU WANT TO DROP ANOTHER COURSE?
ENTER "Y" FOR YES, ANY OTHER CHARACTER FOR NO

A proper response for a course that is not on the current schedule will show the following:

XXXXXX IS NOT ON THE CURRENT SCHEDULE

DO YOU WANT TO DROP ANOTHER COURSE?

A proper response for a course that is "required" will show:

XXXXXX IS A REQUIRED COURSE
DO YOU STILL WISH TO DROP IT?
ENTER "Y" FOR YES, ANY OTHER FOR NO

A "yes" at this point will show the following:

XXXXXX HAS BEEN DROPPED FROM THE SCHEDULE

A "no" will show you this:

XXXXXX WILL REMAIN ON THE SCHEDULE

DO YOU WANT TO DROP ANOTHER COURSE?
ENTER "Y" FOR YES, ANY OTHER CHARACTER FOR NO

A proper response for a non-required course that is on the current schedule will show you the following:

XXXXXX HAS BEEN DROPPED FROM THE SCHEDULE

DO YOU WANT TO DROP ANOTHER COURSE?
ENTER "Y" FOR YES, ANY OTHER CHARACTER FOR NO

A "yes" response will generate the question "which course to drop"?
A "no" response will take you back to the Main Menu.

5.5

MAIN MENU SELECTION E

ENTER "E" TO SELECT AN EMPHASIS AREA

An "e" will cause the following menu to appear:

```
************ SELECT EMPHASIS MENU ***************
*                                                 *
*  SELECT AN EMPHASIS AREA:                       *
*                                                 *
*  ENTER "C" COMPUTER CENTER AND                  *
*                              NETWORK OPERATIONS *
*  ENTER "I" INFORMATION AND COMPUTER NETWORKS    *
*  ENTER "T" TACTICAL SYSTEMS                     *
*  ENTER "D" DECISION SUPPORT SYSTEMS             *
*                                                 *
************ SELECT EMPHASIS MENU ***************
```

An improper response to this menu will merely cause the same menu to reappear until a proper resonse is entered. A proper response will generate the following statement and question:

YOU HAVE SELECTED "XXXXX XXXXX" AS YOUR EMPHASIS AREA

DO YOU WANT TO LIST EMPHASIS AREA COURSES NOW? (Y/N)

A "yes" response at this point will take you to the "list" routine, followed by the question below.

A "no" response will immediately generate the following question:

DO YOU WANT TO ADD EMPHASIS AREA COURSES NOW? (Y,N)

A "yes" response here will take you into the "add a course" routine and then return you to the Main Menu. A "no" response will immediately return you to the Main Menu.

5.6

MAIN MENU SELECTION L

ENTER "L" TO LIST COURSES AVAILABLE

An "1" will cause the following menu to appear:

```
****************** LIST MENU ******************
*                                               *
*  ENTER:                                       *
*    "Q" TO QUIT LISTING                        *
*    "A" TO SEE ALL ADMIN SCIENCE COURSES       *
*    "C" TO SEE ALL COMPUTER SCIENCE COURSES    *
*    "3" TO SEE ALL 3000 LEVEL COURSES          *
*    "4" TO SEE ALL 4000 LEVEL COURSES          *
*    "E" TO SEE ALL EMPHASIS AREA COURSES       *
*    "R" TO SEE ALL REQUIRED COURSES            *
*    "I" TO SEE DATA ON AN INDIVIDUAL COURSE    *
*    "B" TO SEE ALL COURSES IN THE DATA BASE    *
*    "N" TO SEE ALL COURSE "NAMES/TITLES"       *
*    "D" TO SEE COURSE DESCRIPTIONS             *
*                                               *
****************** LIST MENU ******************
```

5.6.1/.2/.3/.4/.5/.6

LIST MENU SELECTION ----------

Character entries of A,C,3,4,R & B are self explainatory. These entries will generate screen output in the selected category in the following format:

				(1 = YES / 0 = NO)			
COURSE	HOURS	CURRIC	PREREQ	FALL/	WINTER/	SPRING/	SUMMER
******	*****	******	******	****	******	******	******
CM3001	4	AS	MN2155	0	0	0	1
CM3002	4	AS	CM3001	1	0	0	0

Following the course listing, ALT PA2 will return you to the listing menu.

5.6.7

LIST MENU SELECTION ----------

An entry of "e" while at the listing menu will generate the following:

SELECT WHICH EMPHASIS AREA COURSES YOU WISH TO LOOK AT:

ENTER:
"C" FOR COMPUTER CENTER AND NETWORK OPERATIONS
"I" FOR INFORMATION AND COMPUTER NETWORKS
"T" FOR TACTICAL SYSTEMS
"D" FOR DECISION SUPPORT SYSTEMS

An improper response at this time will cause the same question to reappear until a proper response is entered. A proper response will show the following for the appropriate area:

YOU HAVE SELECTED "XXXXXX XXXXX"

THE FOLLOWING COURSE IS REQUIRED FOR THIS EMPHASIS AREA:

(1 = YES / 0 = NO)

COURSE	HOURS	CURRIC	PREREQ	FALL	WINTER	SPRING	SUMMER
XXXXXX	X	XX	XXXXXX	0	1	0	0

CHOOSE THREE (3) OF THE FOLLOWING COURSES AS ELECTIVES WITHIN THIS EMPHASIS AREA:

COURSE	HOURS	CURRIC	PREREQ	FALL	WINTER	SPRING	SUMMER
XXXXXX	X	XX	XXXXXX	0	1	0	1

Following the course listing, ALT PA2 will return you to the listing menu.

5.6.8

LIST MENU SELECTION ----------

An entry of "i" while at the listing menu will generate the following:

ENTER THE SIX CHARACTER CODE OF THE COURSE TO LOOK AT

This refers to the two-letter course code and the four-digit course number (e.g. IS2000, MN4105, etc.). An incorrect entry or an entry for a course not resident in the database will bring about the following response (and return you to the listing menu)

XXXXXX IS NOT LOCATED IN THE DATABASE

A proper response will give you the following information on the appropriate course:

				(1 = YES / 0 = NO)			
COURSE	HOURS	CURRIC	PREREQ	FALL/WINTER/SPRING/SUMMER			
******	*****	******	******	****	******	******	*****
XXXXXX	X	XX	XXXXXX	0	1	0	0

Following the course listing, ALT PA2 will return you to the listing menu.

5.6.9

LIST MENU SELECTION ----------

An entry of "n" while at the listing menu will generate a list of course names in the following format:

```
COURSE   COURSE NAME                        (LECTURE/LAB HOURS)
******   ***********                        ********************

CM3001 - ECON EVAL OF TELECOMM SYSTEMS I ------ (4-0)
CM3002 - ECON EVAL OF TELECOMM SYSTEMS II ----- (4-0)
```

Following the course listing, ALT PA2 will return you to the listing menu.

5.6.10

LIST MENU SELECTION ----------

An entry of "d" while at the listing menu will generate the following:

ENTER THE SIX CHARACTER CODE OF THE COURSE TO LOOK AT

This refers to the two-letter course code and the four-digit course number (e.g. IS2000, MN4105, etc.). An incorrect entry or an entry for a course not resident in the database will bring about the following response (and return you to the listing menu):

XXXXXX IS NOT LOCATED IN THE DATABASE

DO YOU WISH TO SEE ANY OTHER COURSE DESCRIPTIONS?
ENTER Y FOR YES / ANY OTHER CHARACTER FOR NO

A proper response will generate the following type of response:

IS4185 - DECISION SUPPORT SYSTEMS ---------- (4-0)
The application and design of computer-based information systems for management planning, control and operations
PREREQUISITES: MN2155, MN3105, OS3101, IS2000

Following the course description, ALT PA.. will generate the following question:

DO YOU WISH TO SEE ANY OTHER COURSE DESCRIPTIONS?
ENTER Y FOR YES / ANY OTHER CHARACTER FOR NO

A "yes" response will regenerate the question "which course to see?"
A "no" response will return you to the listing menu.

5.6.11
LIST MENU SELECTION ----------

An entry of "q" while at the listing menu will return you to Main Menu

---

5.7

MAIN MENU SELECTION U

ENTER "U" TO SEE YOUR UPDATED SCHEDULE AND TO SEE YOUR STATUS ON FULFILLMENT OF MINIMUN REQUIREMENTS

A "u" will generate the output concerning the following topics:

1) courses scheduled per term (number/name/hours)
2) cumulative statistics on:
    a) admin science hours
    b) computer science hours
    c) graduate level hours (3000 & 4000)
    d) 4000 level hours
    e) total hours
3) have enough courses been scheduled per term
4) validated courses
5) required courses not scheduled
6) is an emphasis area selected?
   if so: a) has the required course been scheduled
          b) have 3 electives been scheduled
7) are all prerequisites met?
    a) if not, when must they be scheduled

At various points throughout the display of this information, you will be required to enter ALT PA2 to see the next screenful of information and to return you to the Main Menu.

SAMPLE OUTPUT FROM MAIN MENU SELECTION U

```
****************** CLASSES FOR QTR 1 ********************
CS2970 - STRUCTURED PROGRAMMING WITH PASCAL ------- (5-0)
IS2000 - INTRODUCTION TO COMPUTER MANAGEMENT ------ (3-0)
MN2155 - ACCOUNTING FOR MANAGEMENT ---------------- (4-0)
MN3105 - ORGANIZATIONAL SYSTEMS II ---------------- (4-0)

****************** CLASSES FOR QTR 2 ********************
CS3010 - COMPUTING DEVICES AND SYSTEMS ------------ (4-0)
IS2100 - INFORMATIONS SYSTEMS LABORATORY ---------- (0-2)
IS3170 - ECON EVALUATION OF INFORMATION SYSTEMS I - (4-0)
MN3307 - A.D.P. ACQUISITION ----------------------- (4-0)
OS3101 - STATISTICAL ANALYSIS FOR MANAGEMENT ------ (5-0)

****************** CLASSES FOR QTR 3 ********************
CS3030 - OPERATING SYSTEMS STRUCTURES ------------- (4-0)
IS3171 - ECON EVALUATION OF INFO SYSTEMS II ------- (4-0)
OS3004 - OPS RESEARCH FOR COMPUTER SYS MANAGERS --- (5-0)

****************** CLASSES FOR QTR 4 ********************
CS3020 - SOFTWARE DESIGN -------------------------- (3-2)
IS3502 - COMPUTER COMMUNICATIONS AND NETWORKS ----- (4-0)

****************** CLASSES FOR QTR 5 ********************
IS4183 - APPLICATIONS OF DATABASE MGMT SYSTEMS ---- (4-0)
MN4154 - FINANCIAL MANAGEMENT IN THE ARMED FORCES - (4-0)

****************** CLASSES FOR QTR 6 ********************
IS4182 - INFORMATION SYSTEMS MANAGEMENT ----------- (4-0)
IS4200 - SYSTEM ANALYSIS AND DESIGN --------------- (4-0)
```

SAMPLE OUTPUT (CONTINUED)

```
CUMULATIVE STATISTICS AFTER QTR 6
**********************************
   ADMIN SCIENCE HOURS      =  45
   COMPUTER SCIENCE HOURS   =  17
   GRADUATE LEVEL HOURS     =  58
   4000 LEVEL HOURS         =  16
   TOTAL HOURS              =  72

STATUS OF MINIMUM REQUIREMENT FULFILLMENT
*****************************************
YOU HAVE NOT SCHEDULED ENOUGH CLASSES FOR QTR 3
YOU WILL NEED APPROVAL TO TAKE LESS THAN THE MINIMUM
OF 4 COURSES THIS TERM

YOU HAVE NOT SCHEDULED ENOUGH CLASSES FOR QTR 4,5 & 6
YOU WILL NEED APPROVAL TO TAKE LESS THAN THE MINIMUM
OF 3 COURSES IN THOSE TERMS

VALIDATED COURSES ********************************
      NONE

THE FOLLOWING REQUIRED COURSES HAVE NOT BEEN SCHEDULED
OR VALIDATED:       NONE

AN EMPHASIS AREA HAS NOT BEEN SELECTED YET

ALL PREREQUISITES HAVE BEEN SCHEDULED APPROPRIATELY

------------------------------------------------------------
```

5.8

MAIN MENU SELECTION P

ENTER "P" TO SEE IF PREREQUISITES HAVE BEEN MET
A "p" will show you if all prerequisites have been met
and when deficiencies must be scheduled.

ALT PA2 will return you to the Main Menu.

5.9

MAIN MENU SELECTION Q

ENTER "Q" TO QUIT
A "q" terminates the program and show you the following:

```
...execution ends
File 'SKEDULER': 2696 lines; no diagnostics
110497 bytes of object code generated
5611 statements executed
191192 bytes of memory requested during compilation
6496 bytes returned before execution
100680 bytes requested during execution
```

****

****

6.

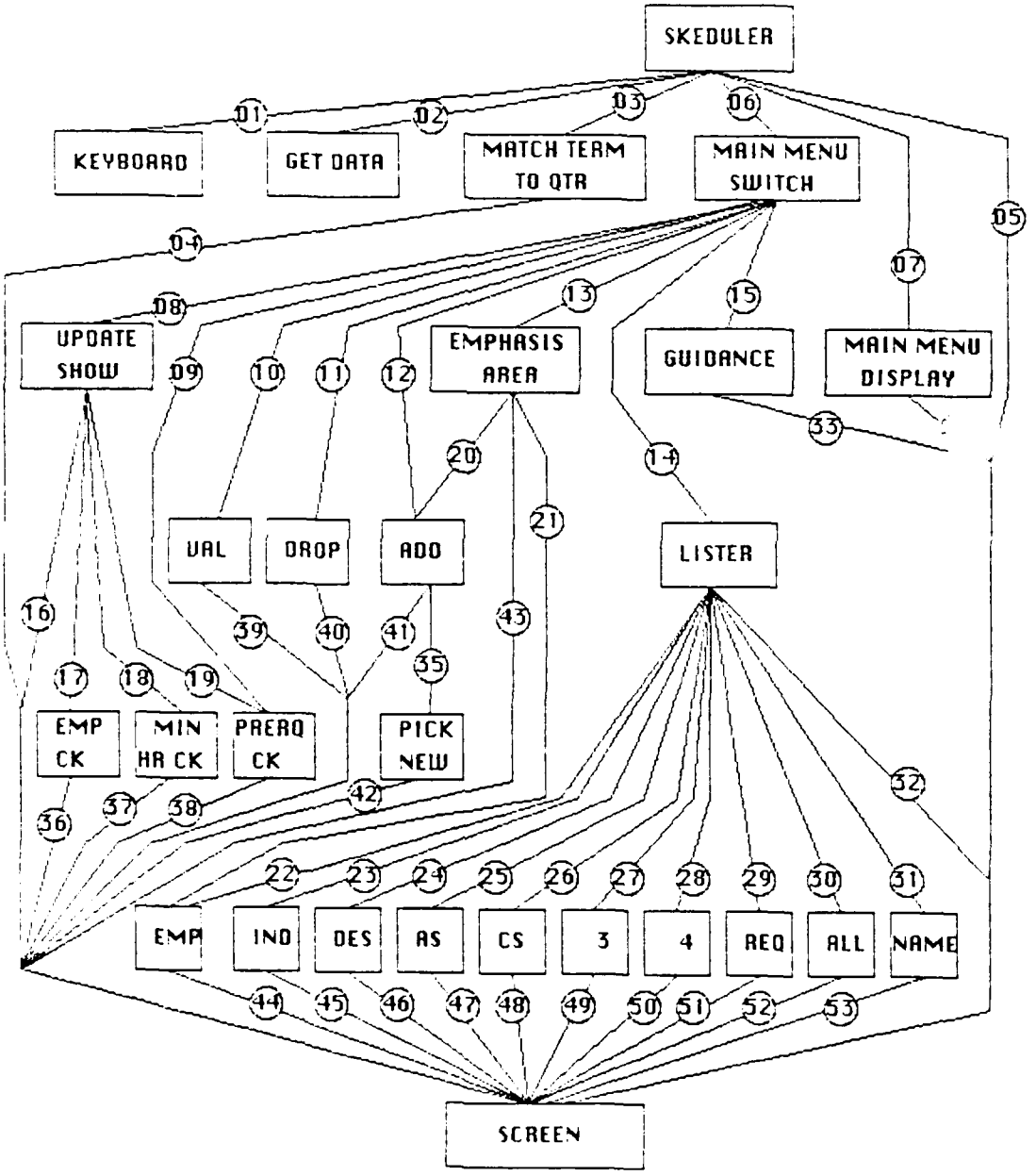
OBTAINING A HARD COPY OF YOUR SCHEDULE

A copy of your latest session on "SKEDULER" will be resident on your a-disk under the file name "SKEDULER LISTING". That file can be edited in XEDIT and printed on the VM printer. (Remember that you will have to "Define Storage 1m" prior to entering XEDIT.)

********** ********

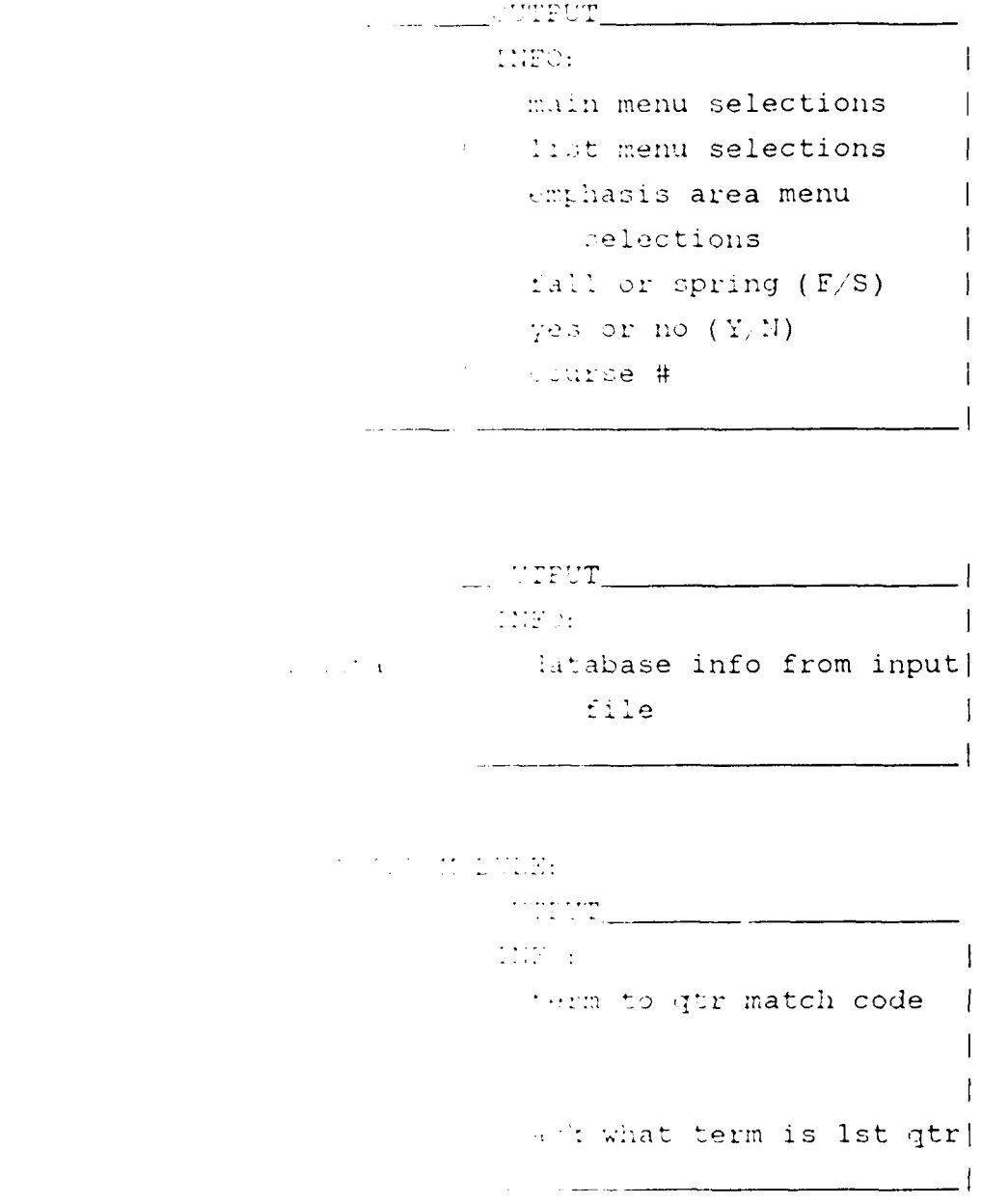
APPENDIX E

STRUCTURE CHART



[illegible] CHART INFORMATION AND CONTROL FLOWS

[illegible]



SKEDULER CONTROL MODULE:

	INPUT	OUTPUT
		INFO:
5.		echo input
6.		main menu selections
		list menu selections
		emphasis area menu selections
		yes or no (Y/N)
		course #
7.		CTRL:
		call main menu display

MAIN MENU SWITCH MODULE:

	INPUT	OUTPUT
		CTRL:
		main menu switch code:
8.		U
9.		P
		INFO:
10.		yes or no (Y/N)
		course #
		CTRL:
		main menu switch code:
		V
11.		D
		INFO:
12.		yes or no (Y/N)
		course #
		term #
		CTRL:
		main menu switch code:
		A
13.		E
		INFO:
14.		yes or no (Y/N)
		course #
		list menu selection
		emphasis area menu selection
		CTRL:
		main menu switch code:
		L
15.		H

UPDATE_SHOW MODULE:

	INPUT	OUTPUT
		INFO:
16.		status of required courses
		list of validated courses
		current schedule
		summary of hours
		CTRL:
17.		call emphasis check
		INFO:
18.		AS,CS,grad,4000 lvl hrs
		CTRL:
19.		call min hrs check
		call prereq check

EMPHASIS_AREA (SELECTION) MODULE:

	INPUT	OUTPUT
		CTRL:
20.		call add_a_course mod
21.		call listemp mod

LIST MENU SWITCH MODULE:

	INPUT	OUTPUT
		CTRL:
22.		list menu switch code(E)
		emphasis menu selection
		INFO:
23.		course #
		CTRL:
		list menu switch code(I)
		INFO:
24.		course #
		CTRL:
		list menu switch code:
		I
25.		A
26.		C
27.		3
28.		4
29.		R
30.		B
31.		N
		INFO:
32.		list menu display

GUIDANCE MODULE:

	INPUT	OUTPUT
		INFO:
33.		program information

MAIN MENU DISPLAY MODULE:

	INPUT	OUTPUT
		INFO:
34.		main menu display

PICK_A_NEW_TERM MODULE:

	INPUT	OUTPUT
	INFO:	CTRL:
35.	yes or no (Y/N)	done
		override data base

SCREEN MODULE:

	INPUT	OUTPUT
	INFO:	
36.	status of emp area courses	
37.	min hour status	
38.	status of prerequisites	
39.	ask for course #	
	tell crs not in database	
	validate another course?	
	show validated course	
40.	ask for course #	
	tell crs not in database	
	tell crs not on schedule	
	tell crs is required, still want to drop?	
	tell crs will remain, drop another?	
	tell crs was dropped	
	drop another?	

SCREEN MODULE (CONTINUED):

	INPUT	OUTPUT
	INFO:	
41.	ask for course #	
	tell crs not in database	
	tell crs is validated still want to add?	
	tell crs already scheduled, want to move it?	
	show when crs offered ,ask when to add it?	
	tell crs not offered when asked for	
	say input must be 1-6	
	tell crs was added	
	add another?	
42.	ask: want to add crs in another term	
	override database?	
43.	show emp area menu	
	tell which area selected	
	ask if want to list emp area crs now	
	ask if want to add emp area crs now	
44.	which emp area to list?	
	show crs for area selected	

SCREEN MODULE (CONTINUED):

	INPUT	OUTPUT
	INFO:	
45.	ask for course #	
	tell crs not in database	
	show selected course	
46.	ask for course #	
	tell crs not in database	
	show sel'd course description	
	see another?	
47.	show AS courses	
48.	show CS courses	
49.	show 3000 level crs	
50.	show 4000 level crs	
51.	show required courses	
52.	show all courses	
53.	show all course names	

APPENDIX G

MODULE DESCRIPTIONS

TABLE OF CONTENTS

MODULE	PAGE
ADD_A_COURSE	92
DESCRIBE	94
DROP_A_COURSE	95
EMP_CHECK	97
EMPHASIS_AREA	98
GET_DATA	99
GUIDANCE	101
KEYBOARD	101
LIST3	102
LIST4	102
LISTALL	103
LISTAS	103
LISTCS	104
LISTEMP	105
LISTER (TRANSACTION DISPATCHER MODULE)	106
LISTIND	108
LIST_REQUIRED_COURSES	109
MAIN_MENU_DISPLAY	109
MAIN MENU SWITCH (DISPATCHER INTERNAL TO CTRL MOD)	116
MATCH_TERM_TO_QTR	110
MIN_HRS_CHECK	111
NAMES	112
PICK_A_NEW_TERM (INTERNAL TO ADD_COURSE)	113
PREREQUISITE_CHECK	114
SCREEN	115
SKEDULER CONTROL MODULE (TRANSACTION CENTER)	116
UPDATE_SHOW	118
VALIDATE	119

MODULE DESCRIPTION

1. NAME: ADD_A_COURSE (ADD)

2. PROSE DESCRIPTION: This module is called by the user by selecting main menu option "A". This module first asks the user which course to add (course #). Lower case input is converted to upper case if necessary. The data base is searched for the selected course. If the course is not found, the user is notified. If the If the course has been validated, the user is notified. He can override the validation and add it back. If the course has been scheduled already, the user is notified. He can override the previous schedule slot and move the course. This module then shows the user when the course is offered (both academic quarter and student term - this is where the "Match_Term_To_Qtr" data is used). This module then asks the user when he wishes to schedule the course. The module determines if the user has asked for the course in a quarter that it is offered (according to the data base). If not, "Pick_A_New_Term" is called in which the user may select a new term or override the data base. After all of these checks, the course is added if the user so desires. Then the user is asked if he wishes to continue adding. If so, the above process is repeated. If not, the user goes back to the main menu.

3. PROCEDURAL DESCRIPTION:

```
Repeat (until thru)
  Ask for input (course #)
  Convert lower case to upper case input
  Search data base for selected course
  If not found then
    Notify the user
```

```
    If found but validated then
      Ask user if he still wants to add
      If no then
        drop to *
    If found but already scheduled then
      Ask user if wants to change the courses location
        on schedule
      If no then
        drop to *
    Show user when course is offered
    Ask which term to add course in
    Read input
    Determine if course is offered when asked for
    If not then
      Call "Pick_A_New_Term"
    Else
      Add the course
  * Ask the user if he wishes to continue adding crs,s
    If no then
      Thru := true
  Until thru
```

4. INTERFACES:

INPUT FROM:	MAIN MENU SWITCH (w/in SKEDULER CONTROL)
	MATCH_TERM_TO_QTR (global data)
	EMPHASIS_AREA (select)
	PICK_A_NEW_TERM (internal to ADD_A_COURSE
OUTPUT TO:	SCREEN
	PICK_A_NEW_TERM (internal to ADD_A_COURSE

MODULE DESCRIPTION

1. NAME: DESCRIBE

2. PROSE DESCRIPTION: The user can call this module by selecting option "D" on the list menu. The user is then prompted for a course #. Upon proper entry of the course #, the user is shown the full description of the selected course.

3. PROCEDURAL DESCRIPTION:

```
Prompt user for course #
Convert lower case input to upper case
Search data base for entered course #
If found then
  Show description of the selected course
If not found then
  Tell user selected crs is not in the data base
```

4. INTERFACES:

INPUT FROM: LISTER (List Menu Switch)

OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: DROP_A_COURSE (DROP)

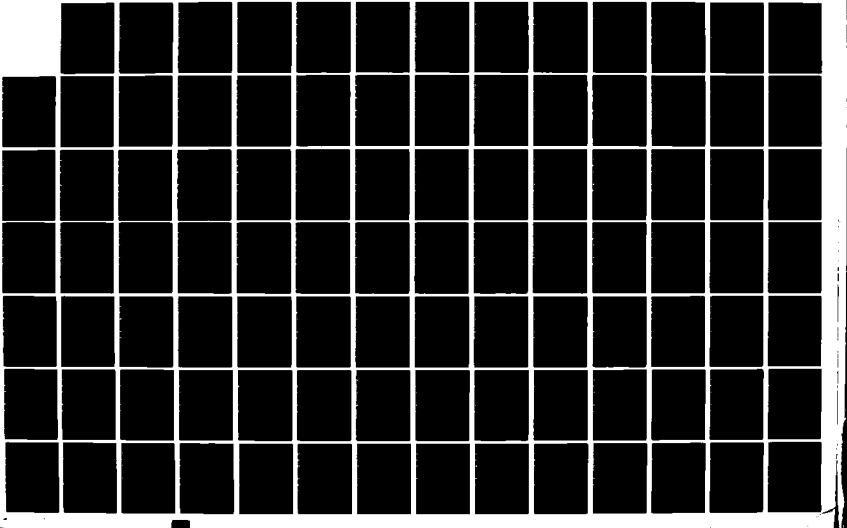
2. PROSE DESCRIPTION: This module is called by the user by selecting main menu option "D". This module first asks the user which course to drop (course #). Lower case input is converted to upper case if necessary. The data base is searched for the selected course. If the course is not found, the user is notified. If the course is found but not scheduled, the user is notified. If the course is required, the user is asked if he still wishes to drop it. If so, the course is dropped. If the course is found and not required it is dropped automatically. Then the user is asked if he wishes to continue dropping courses. If so, the above process is repeated. If not, the user goes back to the main menu.

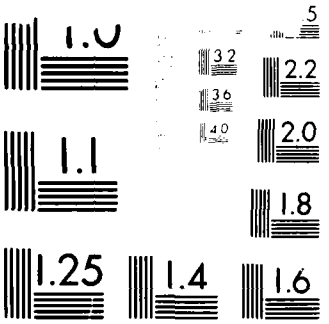
3. PROCEDURAL DESCRIPTION:

```
Repeat (until thru)
  Ask for input (course #)
  Convert lower case to upper case input
  Search data base for selected course
  If found then
    If scheduled then
      If required then
        Ask if user still wants to drop it
        If yes then
          Drop the course (change term code to "0")
      Else
        Drop the course (change term code to "0")
    Else
      Tell the user course not scheduled
  Else
```

AD-A168 843 A MECHANIZED DECISION SUPPORT SYSTEM FOR ACADEMIC SCHEDULING(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S M FENSTERMACHER MAR 86 2/3

UNCLASSIFIED F/G 5/9 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
32
36
40
5

```
        Tell the user the course is not in the data base
      Ask the user if he wishes to continue
      If no then
        Thru := true
    Until thru
```

4. INTERFACES:

   INPUT FROM: MAIN MENU SWITCH (w/in SKEDULER CONTROL)

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: EMP_CHECK

2. PROSE DESCRIPTION: This module is called from "Update_Show". Its function is to check if an emphasis area has been selected and if so, to tell the user if the required emphasis area course and sufficient elective emphasis area courses have been scheduled. (a global emphasis area code is assigned within "Emphasis_Area" upon emp area selection - this global variable is checked within this module to determine if an area has been selected)

3. PROCEDURAL DESCRIPTION:

```
Check EMPCODE (global variable)
If blank then
  Tell user no emphasis area has been selected yet
If EMPCODE assigned then
  If emp crs reqd for selected area not scheduled then
    Tell the user
  Count the # of emp area electives scheduled
    If < 3 then
      Tell the user he needs to schedule at least 3
```

4. INTERFACES:

INPUT FROM: UPDATE_SHOW

OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: EMPHASIS_AREA (select)

2. PROSE DESCRIPTION: This module is called by the user by selecting main menu option "E". The user is shown a menu of options. Upon selection, the EMPCODE (global variable) is set accordingly and the user is asked if he wishes to see emphasis area courses. If so, he is transferred to "Listemp". Upon completion of the list sequence or if he did not desire to see the courses, the user is asked if he wishes to add any courses. If so, he is transferred to the "ADD" module. Upon completion of the add sequence or if he did not wish to add a course, the user goes back to the main menu.

3. PROCEDURAL DESCRIPTION:

```
Show emphasis area option menu
Read user input
Update EMPCODE according to input
Ask to list
  If yes then
    Call "Listemp"
Ask to add
  If yes then
    Call "Add_a_course" (ADD)
```

4. INTERFACES:

   INPUT FROM: MAIN MENU SWITCH (w/in SKEDULER CONTROL)

   OUTPUT TO: SCREEN
   LISTEMP
   ADD (ADD_A_COURSE)

MODULE DESCRIPTION

1. NAME: GET_DATA

2. PROSE DESCRIPTION: Draw data from the input file and load it into the program record structure "data file".

3. PROCEDURAL DESCRIPTION:

Data is to be loaded in two sets:

1) SET 1 - COURSE DATA (FIRST 57 ENTRIES)

COL(S)	CONTENTS	
1- 2	COURSE ID	
4- 7	COURSE NUMBER	
9	HOURS	
11	TERM ASSIGNED	(PRESET SCHEDULE)
		(= ACTUAL TERM PLUS 1)
		(1 = VALIDATED)
13-14	CURRICULUM	
16-17	PREREQUISITE #1 -	COURSE ID
19-23		COURSE NUMBER
24-25	PREREQUISITE #2 -	COURSE ID
27-30		COURSE NUMBER
32-33	PREREQUISITE #3 -	COURSE ID
35-38		COURSE NUMBER
40-41	PREREQUISITE #4 -	COURSE ID
43-46		COURSE NUMBER
48	REQUIRED COURSE CODE	(1=YES / 0=NO)
50	OFFERED IN THE: FALL	(1=YES / 0=NO)
52	WINTER	(" / ")
54	SPRING	(" / ")
56	SUMMER	(" / ")
	EMPHASIS AREA COURSES:	
58	LSS	(2 = REQD / 1=ELECTIVE / 0=NA)
60	TS	(" / " / ")
62	ICN	(" / " / ")
64	CCNO	(" / " / ")

2) SET 2 - COURSE DESCRIPTIONS (2nd 57 ENTRIES)
(EACH ENTRY CONSISTS OF A 10 LINE BLOCK)

COL(S)	CONTENTS
1-73	COURSE NUMBER, NAME, LECTURE/LAB HOURS, DESCRIPTION, PREREQUISITES

4. INTERFACES:

INPUT FROM: SKEDULER CONTROL MODULE

OUTPUT TO: SKEDULER CONTROL MODULE

MODULE DESCRIPTION

1. NAME: GUIDANCE

2. PROSE DESCRIPTION: The user can call this module by selecting main menu option "H". When this module is called, the user is shown help instructions and general information on the program.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:

   INPUT FROM: SKEDULER CONTROL

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: KEYBOARD

2. PROSE DESCRIPTION: Standard IBM 3278 keyboard for all interactive data enteries to the "SKEDULER" program

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:

   OUTPUT TO: SKEDULER CONTROL MODULE

MODULE DESCRIPTION

1. NAME: LIST3

2. PROSE DESCRIPTION: The user can call this module by selecting option "3" on the list menu. When this module is called, the user is shown scheduling data on all 3000 level courses.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:
   INPUT FROM: LISTER (List Menu Switch)

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: LIST4

2. PROSE DESCRIPTION: The user can call this module by selecting option "4" on the list menu. When this module is called, the user is shown scheduling data on all 4000 level courses.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:
   INPUT FROM: LISTER (List Menu Switch)

   CUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: LISTALL

2. PROSE DESCRIPTION: The user can call this module by selecting option "B" on the list menu. When this module is called, the user is shown scheduling data on all courses in the data base.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:

   INPUT FROM: LISTER (List Menu Switch)

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: LISTAS

2. PROSE DESCRIPTION: The user can call this module by selecting option "A" on the list menu. When this module is called, the user is shown scheduling data on all Admin Science courses.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:

   INPUT FROM: LISTER (List Menu Switch)

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: LISTCS

2. PROSE DESCRIPTION: The user can call this module by selecting option "C" on the list menu. When this module is called, the user is shown scheduling data on all Computer Science courses.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:

   INPUT FROM: LISTER (List Menu Switch)

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: LISTEMP

2. PROSE DESCRIPTION: The user can call this module by selecting option "E" on the list menu or by answering appropriate questions from within module "Emphasis Area". When this module is called, the user is shown a menu of emphasis areas from which to choose. Upon selection from that menu, the user is shown the scheduling data on all courses that are required by or used as electives for the chosen emphasis area.

3. PROCEDURAL DESCRIPTION:

```
Repeat (until good)
  Show emphasis area selection menu
  Read input from user
  Error check the input
  If input OK then
    good := true
Until good
Show user appropriate courses based on the input
  INPUT___SHOW
    C     Computer Center and Network Operations
    I     Information and Computer Networks courses
    T     Tactical Systems courses
    D     Decision Support Systems Courses
```

4. INTERFACES:

INPUT FROM: LISTER (List Menu Switch)
EMPHASIS AREA (selection)

OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: LISTER (LIST MENU SWITCH)

2. PROSE DESCRIPTION: The user can call this module by selecting main menu option "L". When this module is called, the user is shown a Listing Menu from which to select his next move. The user is then routed to the appropriate subroutine based on his list menu option selection. This is a transaction dispatcher module.

3. PROCEDURAL DESCRIPTION:

```
Repeat (until move)
  Repeat (until good)
    Show list menu
    Read selection input
    Error check the above input
    If input OK then
      good := true
  Until good
  Route the user to the appropriate list module
    INPUT___MODULE_____________ACTION
      Q      move := true      leave the listing module
      A      LISTAS            list AS courses
      C      LISTCS            list CS courses
      3      LIST3             list 3000 level courses
      4      LIST4             list 4000 level courses
      E      LISTEMP           list emphasis area crs
      R      LIST_REQUIRED_COURSES
                               list required courses
      I      LISTIND           list an individual crs
      B      LISTALL           list all courses
      N      NAMES             list names of all courses
      D      DESCRIBE          list a course description
Until move
```

4. INTERFACES:

INPUT FROM: MAIN MENU SWITCH (w/in SKEDULER CONTROL)

OUTPUT TO: SCREEN
LISTEMP
LISTIND
LISTAS
LISTCS
LISTALL
LIST_REQUIRED_COURSES
LIST3
LIST4
NAMES
DESCRIBE

MODULE DESCRIPTION

1. NAME: LISTIND

2. PROSE DESCRIPTION: The user can call this module by selecting option "I" on the list menu. The user is then prompted for a course #. Upon proper entry of the course #, the user is shown the scheduling data on the selected course.

3. PROCEDURAL DESCRIPTION:

```
Prompt user for course #
Convert lower case input to upper case
Search data base for entered course #
If found then
  Show data on the selected course
If not found then
  Tell user selected course is not in the data base
```

4. INTERFACES:

INPUT FROM: LISTER (List Menu Switch)

OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: LIST_REQUIRED_COURSES

2. PROSE DESCRIPTION: The user can call this module by selecting option "R" on the list menu. When this module is called, the user is shown scheduling data on all required courses.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:
   INPUT FROM: LISTER (List Menu Switch)

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: MAIN_MENU_DISPLAY

2. PROSE DESCRIPTION: Called from the "Skeduler Control" module to display the main menu options to the user.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:
   INPUT FROM: SKEDULER CONTROL

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: MATCH_TERM_TO_QTR

2. PROSE DESCRIPTION: This module takes input from the user to corrolate the student term sequence (i.e. 1-6) to the academic quarter sequence (i.e. fall,winter,spring,summer). This info is used in the "Add" module to show students when courses are offered.

3. PROCEDURAL DESCRIPTION:

Get input from user as to first term
Based on the above input,
assign academic qtrs to student term #s
If 1st term = fall then

TERM	QUARTER
1	FALL
2	WINTER
3	SPRING
4	SUMMER
5	FALL
6	WINTER

If 1st term = spring then

TERM	QUARTER
1	SPRING
2	SUMMER
3	FALL
4	WINTER
5	SPRING
6	SUMMER

4. INTERFACES:

INPUT FROM: SKEDULER CONTROL MODULE

OUTPUT TO: SKEDULER CONTROL MODULE
ADD (via global data)

MODULE DESCRIPTION

1. NAME: MIN_HRS_CHECK

2. PROSE DESCRIPTION: The # of hours scheduled in the following areas (AS, CS, grad, 4000 lvl, total) are passed to this module from "Update_Show". Those hours are compared to standard minimums. Shortages are shown to the user. Shortages in classes scheduled per term are also passed from "Update_Show" and shown to the user in this module.

3. PROCEDURAL DESCRIPTION:

Compare AS hours scheduled to min AS hours required
Compare CS hours scheduled to min CS hours required
Compare grad hours scheduled to min grad hours reqd
Compare 4000 lvl hours scheduled to min 4000 lvl hours required
Examine # classes per term
Display all deficiencies to the user

4. INTERFACES:

INPUT FROM: UPDATE_SHOW

OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: NAMES

2. PROSE DESCRIPTION: The user can call this module by selecting option "N" on the list menu. When this module is called, the user is shown the names of all courses in the data base.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:

   INPUT FROM: LISTER (List Menu Switch)

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: PICK_A_NEW_TERM (internal to ADD_A_COURSE)

2. PROSE DESCRIPTION: Called fron "ADD_A_COURSE" when the user tries to add a course in term it is not offered (according to the data base). The user is asked if he wishes to change his term selection or if he wishes to override the data base. His choice is then passed to the "ADD" module for appropriate action.

3. PROCEDURAL DESCRIPTION:

```
Ask user if he wishes to change term selection
If no then
  Ask user if he wishes to override the data base
  If yes then
    Notify "Add" to add the course
  If no then
    Notify "Add" to get another term selection
```

4. INTERFACES:

INPUT FROM: ADD_A_COURSE (ADD)

OUTPUT TO: ADD_A_COURSE (ADD)
SCREEN

MODULE DESCRIPTION

1. NAME: PREREQUISITE_CHECK

2. PROSE DESCRIPTION: This module is called automatically when the user updates his schedule (selection "U" on the main menu). It can also be called directly by the user by selecting option "P" from the main menu. This module searches the data base for scheduled courses. As it finds a scheduled course, it examines the prerequisite fields of that course one at a time. If there is an entry in the prereq field this module then increments through the data base again to see if that course has been scheduled and if so, when it is scheduled for. It then notifies the user of any prerequisites that are not scheduled, scheduled concurrent with the master course, or scheduled after the master course.

3. PROCEDURAL DESCRIPTION:

```
Increment through the data base one course at a time
If course is scheduled then
  For X := 1 to 4 do
    If prerequisite #X not scheduled then
      Notify user
    Else
      If term of prerequisite #X >=
                        term of scheduled course then
        Tell user prerequisite scheduled concurrent
                        or too late
```

4. INTERFACES:

   INPUT FROM: UPDATE_SHOW
   MAIN MENU SWITCH (w/in SKEDULER CONTROL)

   OUTPUT TO: SCREEN

MODULE DESCRIPTION

1. NAME: SCREEN

2. PROSE DESCRIPTION: Standard IBM 3278 Terminal screen for all interactive output.

3. PROCEDURAL DESCRIPTION: none

4. INTERFACES:

INPUT FROM: SKEDULER CONTROL
MATCH_TERM_TO_QTR
UPDATE_SHOW
EMP_CHECK
MIN_HRS_CHECK
PREREQUISITE_CHECK
VALIDATE
DROP_A_COURSE
ADD_A_COURSE
EMPHASIS_AREA
GUIDANCE
MAIN_MENU_DISPLAY
LISTER
LISTEMP
LISTIND
DESCRIBE
LISTAS
LISTCS
LIST3
LIST4
LIST_REQUIRED_COURSES
LISTALL
NAMES

MODULE DESCRIPTION

1. NAME: SKEDULER CONTROL MODULE / MAIN MENU SWITCH

2. PROSE DESCRIPTION: This module coordinates and controls module execution and the flow of information. Contains "Main Menu Switch" module which is a transaction dispatcher.

3. PROCEDURAL DESCRIPTION:

```
Call module "Get Data" to load input file into the
     database
Reset program to receive input from the terminal
Call module "Match Term to Qtr"
Call module "Main Menu Display"
Repeat (until thru)
  Repeat (until good)
    Receive input from terminal
    Error check the input
    If OK then
      good := true
  Until good
  Route user to appropriate module
    INPUT---MODULE
      Q      thru := true
      A      ADD
      D      DROP
      V      VALIDATE
      E      EMPHASIS AREA
      L      LISTER (List Menu Switch)
      U      UPDATE_SHOW
      P      PREREQUISITE_CHECK
      H      GUIDANCE (Help)
Until thru
```

4. INTERFACES:

INPUT FROM: KEYBOARD
GET DATA
MATCH TERM TO QTR

OUTPUT TO: GET DATA
MATCH TERM TO QTR
MAIN MENU SWITCH (internal)
UPDATE_SHOW
PREREQUISITE_CHECK
VALIDATE
ADD
DROP
EMPHASIS AREA (select)
LISTER (List Menu Switch)
GUIDANCE
MAIN MENU DISPLAY
SCREEN

MODULE DESCRIPTION

1. NAME: UPDATE_SHOW

2. PROSE DESCRIPTION: The user can call this module by selecting option "U" on the main menu. The user is shown all of the courses in his preset schedule and all of the modifications he has made to that schedule. The user is also shown the cumulative hours scheduled in five catagories: Admin Science, Computer Science, Graduate level, 4000 level, and total. This module also shows all validated courses as well as the status of required courses. It then calls three other modules (MIN_HRS_CHECK, EMP_CHECK, PREREQUISITE_CHECK) to display other relevant scheduling data.

3. PROCEDURAL DESCRIPTION:

Increment through the data base
Look at scheduled courses only
Count # of AS, CS, 3000 lvl, 4000 lvl hours
Determine if enough classes taken per term
Display scheduled courses by term
Display hours by catagory
Call "Min_Hrs_Check"
Show validated courses
Show status of required courses
Call "Emp_Check"
Call "Prerequisite_Check"

4. INTERFACES:

INPUT FROM: MAIN MENU SWITCH (w/in SKEDULE CONTROL)

OUTPUT TO: SCREEN
MIN_HRS_CHECK
EMP_CHECK
PREREQUISITE_CHECK

MODULE DESCRIPTION

1. NAME: VALIDATE

2. PROSE DESCRIPTION: This module is called by the user by selecting main menu option "V". This module begins by asking the user for the course # he wishes to validate. It then convert lower case input to upper case if necessary. Next it searches the data base for the selected course, notifies the user if it is not found and changes the term code to "1" if it is found. Finally, this module asks the user if he wishes to validate any more courses. If so, the above process is repeated. If not, the module shows the user all validated courses.

3. PROCEDURAL DESCRIPTION:

```
Repeat (until thru)
  Ask for course #
  Convert to upper case
  Search data base for course # selected
  If not found then
    notify user
  Else
    change term code to "1"
  Ask user if he wants to continue
  If no then
    Thru := true
Until thru
Show all validated courses
```

4. INTERFACES:

INPUT FROM: MAIN MENU SWITCH (w/in SKEDULER CONTROL)

OUTPUT TO: SCREEN

APPENDIX H

PROGRAM LISTING

```
(*$S100000*)
PROGRAM SKEDULER (INPUT,OUTPUT);

(*********************************************************************)
(*                                                                   *)
(*   TITLE: SKEDULER-"A KNOWLEDGE BASED DECISION SUPPORT SYSTEM      *)
(*                   FOR CSM COURSE SELECTION  AND  SCHEDULING"      *)
(*  AUTHOR: S. M. FENSTERMACHER                                      *)
(*    DATE: 16 OCTOBER 1985                                          *)
(*                                                                   *)
(*  DESCRIPTION:  THIS PROGRAM IS DESIGNED TO HELP COMPUTER SYSTEM   *)
(*       MGMT (CURRIC #367 - CSM) STUDENTS PREPARE THEIR ACADEMIC    *)
(*       SCHEDULES FOR THEIR ENTIRE STAY AT NPS.  THE USER IS        *)
(*       PROVIDED WITH A PRESET SCHEDULE OF REQUIRED COURSES AND     *)
(*       MUST BUILD ON AND MODIFY THAT SCHEDULE IN ORDER TO MEET     *)
(*       PERSONAL NEEDS AS WELL AS MINIMUM REQUIREMENTS FOR          *)
(*       GRADUATION:                                                 *)
(*             CATEGORY                      MINIMUM                 *)
(*                                                                   *)
(*             ADMIN SCIENCE HOURS              24                   *)
(*             COMPUTER SCIENCE HOURS           16                   *)
(*             GRADUATE LEVEL HOURS             48                   *)
(*             4000 LEVEL HOURS                 12 (OF THE 48 GRAD)  *)
(*             COURSES PER TERM (1-3)            4                   *)
(*             COURSES PER TERM (4-6)            3                   *)
(*                                                                   *)
(*       PLUS:                                                       *)
(*             SCHEDULE OR VALIDATE ALL REQUIRED COURSES             *)
(*             SELECT AN EMPHASIS AREA WITH:                         *)
(*                  A) 1 REQUIRED COURSE                             *)
(*                  B) 3 ELECTIVES                                   *)
(*                                                                   *)
(*       PLUS:                                                       *)
(*             ENSURE THAT PREREQUISTES FOR ALL COURSES ARE MET      *)
(*                                                                   *)
(*       THIS MUST ALL BE DONE WHILE KEEPING IN MIND THAT MOST       *)
(*       COURSES ARE NOT OFFERED EVERY ACADEMIC QUARTER.             *)
(*                                                                   *)
(*       THIS PROGRAM IS DESIGNED TO GIVE THE STUDENT A MINIMUM      *)
(*       AMOUNT OF INFORMATION (TO AVOID OVERLOAD) AND STILL ALLOW   *)
(*       HIM/HER TO MAKE INFORMED DECISIONS ABOUT HIS/HER            *)
(*       ACADEMIC FUTURE.  THIS PROGRAM ALLOWS  STUDENTS TO GENERATE *)
(*       ANY SCHEDULE DESIRED (WITHIN DATA BASE LIMITATIONS)         *)
(*       AND WILL KEEP THEM APPRAISED OF THEIR STATUS                *)
(*       REGARDING FULFILLMENT OF MINIMUM REQUIREMENTS,              *)
(*       PREREQUISITES AND SCHEDULING FEASIBILITIES.                 *)
(*                                                                   *)
(*       ASSUMPTIONS:                                                *)
```

```
     1) STUDENTS WILL BEGIN IN THE FALL OR SPRING ONLY
     2) "EITHER/OR" PREREQUISITES ARE NOT CONSIDERED

LIMITATIONS:
     1) THE DATABASE OF COURSES IS LIMITED TO:
          A) REQUIRED COURSES
          B) EMPHASIS AREA COURSES
          C) PREREQUISITES FOR:
               (1) REQUIRED COURSES
               (2) EMPHASIS AREA COURSES
               (3) PREREQUISITES
     2) NO MATTERS REGARDING A THESIS ARE CONSIDERED

ORIENTATION:
     TERMINAL ORIENTED/INTERACTIVE/MENU DRIVEN

TO OBTAIN HARD COPY OF SCHEDULE CREATED:
   A COPY OF THE LATEST ENTIRE SESSION ON "SKEDULER"
   WILL BE RESIDENT ON YOUR A-DISK UNDER THE FILE
   "SKEDULER LISTING".  THAT FILE CAN BE EDITED IN
   XEDIT AND PRINTED ON THE VM PRINTER.  IF YOUR
   LAST SESSION WAS OF CONSIDERABLE LENGTH, YOU MAY
   HAVE TO "DEFINE STORAGE 1M" PRIOR TO GOING INTO
   XEDIT OR YOU WILL BE TOLD THAT YOUR LISTING FILE
   IS TOO LARGE.

THE DATABASE OF COURSES ( FOLLOWING "$ENTRY" BELOW ):

 1) SET 1 - COURSE DATA ( FIRST 57 ENTRIES)
   COL(S)  CONTENTS

     1- 2   COURSE ID
     4- 7   COURSE NUMBER
        9   HOURS
       11   TERM ASSIGNED   (PRESET SCHEDULE)
                            (= ACTUAL TERM PLUS 1)
                            (1 = VALIDATED)
    13-14   CURRICULUM
    16-17   PREREQUISITE #1 - COURSE ID
    19-23                     COURSE NUMBER
    24-25   PREREQUISITE #2 - COURSE ID
    27-30                     COURSE NUMBER
    32-33   PREREQUISITE #3 - COURSE ID
    35-38                     COURSE NUMBER
    40-41   PREREQUISITE #4 - COURSE ID
    43-46                     COURSE NUMBER
       48   REQUIRED COURSE CODE    (1 = YES / 0 = NO)
       50   OFFERED IN THE: FALL    (1 = YES / 0 = NO)
```

```
(*              52                    WINTER  (    "    /    "    )        *)
(*              54                    SPRING  (    "    /    "    )        *)
(*              56                    SUMMER  (    "    /    "    )        *)
(*                   EMPHASIS AREA COURSES:                                *)
(*              58   DSS    (2 = REQD / 1 = ELECTIVE / 0 = NA)             *)
(*              60   TS     (    "    /        "      /    "    )          *)
(*              62   ICN    (    "    /        "      /    "    )          *)
(*              64   CCNO   (    "    /        "      /    "    )          *)
(*                                                                         *)
(*         2) SET 2 - COURSE DESCRIPTIONS (SECOND 57 ENTRIES)              *)
(*                    (EACH ENTRY CONSISTS OF A 10 LINE BLOCK)             *)
(*           COL(S)  CONTENTS                                              *)
(*                                                                         *)
(*           1-73    COURSE NUMBER, NAME, LECTURE/LAB HOURS,               *)
(*                   DESCRIPTION, PREREQUISITES                            *)
(*                                                                         *)
(*    *$S100000*        : USED ON LINE 1 TO PREVENT STACK OVERFLOW        *)
(*                                                                         *)
(*   PRIOR TO EXECUTION : DEFINE STORAGE 1M                                *)
(*                        I CMS                                            *)
(*                                                                         *)
(*   EXECUTION STATEMENT: PW SKEDULER  (WHILE IN CMS)                      *)
(*                                                                         *)
(*   COMPILE TIME       : 30 SEC (APPROX)                                  *)
(*                                                                         *)
(*************************************************************************)

CONST
   CODELENGTH = 2;
   NUMBLENGTH = 4;
   NR_COURSES = 57;     (* NUMBER OF COURSES IN THE DATABASE *)
   MINGRADHRS = 48;
   MINLVL4HRS = 12;
   MINASHRS   = 24;
   MINCSHRS   = 16;
   MINEMP     =  3;
   SPACE      = "                                                    ";

TYPE
   COURSECODE = PACKED ARRAY (.1 .. CODELENGTH.) OF CHAR;
   COURSENUMB = PACKED ARRAY (.1 .. NUMBLENGTH.) OF CHAR;
   DESCRIBELN = PACKED ARRAY (.1..73.) OF CHAR;
   DSCRIBPARA = PACKED ARRAY (.1..10.) OF DESCRIBELN;

   SCHEDULE   = RECORD
         COURSEID : COURSECODE;
         COURSENR : COURSENUMB;
         HOURS    : INTEGER;
```

```
            TERM      : INTEGER;
            CURRIC    : COURSECODE;
            PREID1    : COURSECODE;
            PRENR1    : COURSENUMB;
            PREID2    : COURSECODE;
            PRENR2    : COURSENUMB;
            PREID3    : COURSECODE;
            PRENR3    : COURSENUMB;
            PREID4    : COURSECODE;
            PRENR4    : COURSENUMB;
            REQD      : INTEGER;
            FALL      : INTEGER;
            WINTER    : INTEGER;
            SPRING    : INTEGER;
            SUMMER    : INTEGER;
            DSS       : INTEGER;
            TS        : INTEGER;
            ICN       : INTEGER;
            CCNO      : INTEGER;
            DESCRIPT  : DSCRIBPARA;
            DROP      : CHAR;
        END; (* END OF RECORD *)

    LIST       = ARRAY (.1 .. NR_COURSES.) OF SCHEDULE;

VAR
    DATA_FILE  : LIST;
    INPT       : TEXT;     (* TO RECEIVE INPUT FROM TERMINAL *)
    ASHRS      : INTEGER;   {*********************************}
    CSHRS      : INTEGER;   {*  THESE VARIABLES ARE USED      *}
    LVL4HRS    : INTEGER;   {*  TO KEEP TRACK OF ACTUAL       *}
    GRADHRS    : INTEGER;   {*  HOURS SCHEDULED               *}
    TOTALHRS   : INTEGER;   {*********************************}
    ASDEF      : INTEGER;  {*****************************************}
    CSDEF      : INTEGER;  {* THESE VARIABLES ARE USED TO RECORD      *}
    GRADDEF    : INTEGER;  {* DEFICIENCIES OF REQUIRED MINIMUM HOURS  *}
    LVL4DEF    : INTEGER;  {*****************************************}
    NR_CLASSES : ARRAY (.2..7.) OF BOOLEAN;  (* QTR CLASS COUNT CHECK *)
    EMPCODE    : CHAR;      (* DESIGNATES WHICH EMPHASIS AREA SELECTED *)
    TERM1      : CHAR;  {************************************}
    TERM2      : CHAR;  {* THESE ARE ASSIGNED IN PROCEDURE  *}
    TERM3      : CHAR;  {* MATCH_TERM_TO_QTR AND ARE USED   *}
    TERM4      : CHAR;  {* TO CHECK AVAILABILITY OF COURSES *}
    TERM5      : CHAR;  {* IN PROCEDURE ADD_A_COURSE        *}
    TERM6      : CHAR;  {************************************}
    TERMCHOICE : CHAR;
    THRU,GOOD  : BOOLEAN;
```

```
   MENU        : CHAR;

{*************************************************************}
{* GET DATA FROM INPUT FILE                                  *}
{*************************************************************}

PROCEDURE GET_DATA(VAR REC_SCH: LIST);

VAR
   LENGTH  : INTEGER;
   LENGTH3 : INTEGER;
   AA      : INTEGER;

BEGIN  (* PROCEDURE *)

   FOR LENGTH := 1 TO NR_COURSES DO
      BEGIN
        WITH REC_SCH(.LENGTH.) DO

          BEGIN

           READ(COURSEID);
           READ(DROP);
           READ(COURSENR);
           READ(HOURS);
           READ(TERM);
           READ(DROP);
           READ(CURRIC);
           READ(DROP);
           READ(PREID1);
           READ(DROP);
           READ(PRENR1);
           READ(DROP);
           READ(PREID2);
           READ(DROP);
           READ(PRENR2);
           READ(DROP);
           READ(PREID3);
           READ(DROP);
           READ(PRENR3);
           READ(DROP);
           READ(PREID4);
           READ(DROP);
           READ(PRENR4);
           READ(REQD);
           READ(FALL);
           READ(WINTER);
           READ(SPRING);
```

```
            READ(SUMMER);
            READ(DSS);
            READ(TS);
            READ(ICN);
            READ(CCNO);
            READLN;

           END;  (* OF WITH *)


       END;  (* OF FOR LOOP *)

   FOR LENGTH3 := 1 TO NR_COURSES DO
     BEGIN (* FOR *)
       WITH REC_SCH(.LENGTH3.) DO
         BEGIN (* WITH *)
           FOR AA := 1 TO 10 DO
            BEGIN (* FOR *)
             READLN(DESCRIPT(.AA.));
             STRCONCAT (DESCRIPT(.AA.),SPACE); (* FOR LEFT JUSTIFY *)
            END;  (* FOR *)
         END;  (* WITH *)
     END;  (* FOR *)

END;  (* OF GET_DATA *)

(*********************************************************************)
(* MATCH TERM #S TO ANNUAL QTR (SEASONS) FOR CLASS OFFERINGS         *)
(* -THE DATABASE IS CURRENTLY SET UP FOR FALL AND SPRING STARTS ONLY *)
(*********************************************************************)

PROCEDURE MATCH_TERM_TO_QTR;

VAR
   OK          : BOOLEAN;
   RESPONSE    : CHAR;

BEGIN  (* PROCEDURE *)

  WRITELN('                                                        ');
  WRITELN('                 ************************************   ');
  WRITELN('                 *   ******************************   *   ');
  WRITELN('                 *   *   WELCOME TO "SKEDULER"    *   *   ');
  WRITELN('                 *   ******************************   *   ');
  WRITELN('                 ************************************   ');

  OK := FALSE;(* INITIALIZE *)
  REPEAT  (* UNTIL OK *)
```

```
    WRITELN('                                                              ');
    WRITELN('                                                              ');
    WRITELN('                                                              ');
    WRITELN('                                                              ');
    WRITELN('                                                              ');
    WRITELN('                                                              ');
    WRITELN('            WHAT QTR OF THE YEAR IS YOUR FIRST TERM IN?       ');
    WRITELN('                                                              ');
    WRITELN('                  ENTER "F" FOR FALL    (OCT-DEC)            ');
    WRITELN('                  ENTER "S" FOR SPRING  (APR-JUN)            ');
    WRITELN;
    READLN(INPT,RESPONSE);
    WRITELN(RESPONSE);          (* ECHO PRINT *)
    WRITELN;
      IF RESPONSE IN (.'F','f','S','s'.) THEN
        OK := TRUE
      ELSE
        BEGIN
          WRITELN('SORRY, WRONG INPUT, TRY AGAIN');
          PAGE;    (* CLEAR SCREEN *)
        END;
    UNTIL OK;  (* END REPEAT LOOP *)

    PAGE;  (* CLEAR SCREEN *)

    IF RESPONSE IN (.'F','f'.) THEN
       BEGIN  (* IF F *)
         TERM1 := 'F';
         TERM2 := 'W';
         TERM3 := 'S';
         TERM4 := 'R';
         TERM5 := 'F';
         TERM6 := 'W';
       END    (* IF F *)
    ELSE
    IF RESPONSE IN (.'S','s'.) THEN
       BEGIN  (* IF S *)
         TERM1 := 'S';
         TERM2 := 'R';
         TERM3 := 'F';
         TERM4 := 'W';
         TERM5 := 'S';
         TERM6 := 'R';
       END    (* IF S *)

END;   (* PROCEDURE *)

(*************************************************************************)
```

```
{* LIST ALL REQUIRED COURSES FOR THE USER                                    *}
{*****************************************************************************}

PROCEDURE LIST_REQUIRED_COURSES;

VAR
   R           : INTEGER;  (* INDEX *)

BEGIN  (* PROCEDURE *)

   WRITE('THE FOLLOWING IS A LIST OF REQUIRED COURSES FOR THE');
   WRITELN(' COMPUTER SYSTEMS MANAGEMENT CIRRICULUM');
   WRITELN;
   WRITE('                                                                  ');
   WRITELN('   ( 1 = YES / 0 = NO )');
   WRITE('COURSE  HOURS  CURRIC  PREREQUISITES            OFFERED: ');
   WRITELN('FALL/WINTER/SPRING/SUMMER');
   WRITE('******  *****  ******  *************                    ');
   WRITELN('**** ****** ****** ******');
   WRITELN;
   FOR R := 1 TO NR_COURSES DO
      BEGIN (* FOR R *)
         WITH DATA_FILE(.R.) DO
            BEGIN (* WITH *)
               IF REQD = 1 THEN
                  BEGIN  (* IF 1 *)
                   WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                   WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                   WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                   WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
                  END;   (* IF 1 *)
            END;  (* WITH *)
      END;  (* FOR R *)
      WRITELN;  (**)
      WRITELN;   (**)
      WRITELN;    (**)
      WRITELN;     (**)
      WRITELN;      (**)
      WRITELN;       (**)
      WRITELN;        (**)
      WRITELN;         {*********************************************}
      WRITELN;         {*  THE PURPOSE OF THESE WRITELN'S IS TO PUSH *}
      WRITELN;         {*  THE LIST MENU TO THE NEXT FULL SCREEN     *}
      WRITELN;         {*********************************************}
      WRITELN;        (**)
      WRITELN;       (**)
      WRITELN;      (**)
      WRITELN;     (**)
```

```
      WRITELN;     (**)
      WRITELN;     (**)
      WRITELN;   (**)

END;    (* PROCEDURE *)

{*************************************************************************}
{* LIST ADMIN SCIENCE COURSES                                            *}
{*************************************************************************}

PROCEDURE LISTAS;

VAR
   D           : INTEGER;     (* INDEX *)

BEGIN  (* PROCEDURE *)

  WRITELN('ADMIN SCIENCE COURSES');
  WRITELN;
  WRITE('                                                              ');
  WRITELN('   ( 1 = YES / 0 = NO )');
  WRITE('COURSE  HOURS  CURRIC  PREREQUISITES            OFFERED: ');
  WRITELN('FALL/WINTER/SPRING/SUMMER');
  WRITE('******  *****  ******  **************                        ');
  WRITELN('**** ****** ****** ******');
  WRITELN;
  FOR D := 1 TO NR_COURSES DO
    BEGIN  (* FOR D *)
      WITH DATA_FILE(.D.) DO
        BEGIN  (* WITH *)
          IF CURRIC = 'AS' THEN
            BEGIN  (* IF AS *)
              WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
              WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
              WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
              WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
            END;    (* IF AS *)
        END;    (* WITH *)
    END;    (* FOR D *)
    WRITELN;  {****************************************}
    WRITELN;  {* THESE WRITELN'S ARE USED TO PUSH THE *}
    WRITELN;  {* LIST MENU TO THE NEXT FULL SCREEN    *}
              {****************************************}
END;    (* PROCEDURE *)


{*************************************************************************}
{* LIST COMPUTER SCIENCE COURSES                                         *}
```

```
(*************************************************************************)

PROCEDURE LISTCS;

VAR
   E           : INTEGER;     (* INDEX *)

BEGIN  (* PROCEDURE *)

  WRITELN('COMPUTER SCIENCE COURSES');
  WRITELN;
  WRITE('                                                              ');
  WRITELN('   ( 1 = YES / 0 = NO )');
  WRITE('COURSE  HOURS  CURRIC  PREREQUISITES            OFFERED: ');
  WRITELN('FALL/WINTER/SPRING/SUMMER');
  WRITE('******  *****  ******  ************                        ');
  WRITELN('**** ****** ****** ******');
  WRITELN;
  FOR E := 1 TO NR_COURSES DO
    BEGIN  (* FOR E *)
      WITH DATA_FILE(.E.) DO
        BEGIN  (* WITH *)
          IF CURRIC = 'CS' THEN
            BEGIN  (* IF CS *)
              WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
              WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
              WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
              WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
            END;    (* IF CS *)
        END;    (* WITH *)
    END;    (* FOR E *)
    WRITELN;  {****************************************}
    WRITELN;  {* THESE WRITELN'S ARE USED TO PUSH THE *}
    WRITELN;  {* LIST MENU TO THE NEXT FULL SCREEN    *}
              {****************************************}
END;    (* PROCEDURE *)

{*************************************************************************}
{* LIST 3000 LEVEL COURSES                                               *}
{*************************************************************************}

PROCEDURE LIST3;

VAR
   F           : INTEGER;    (* INDEX *)

BEGIN  (* PROCEDURE *)
```

```
    WRITELN('3000 LEVEL COURSES');
    WRITELN;                                                              ');
    WRITE('
    WRITELN('   ( 1 = YES / 0 = NO )');
    WRITE('COURSE  HOURS  CURRIC  PREREQUISITES            OFFERED: ');
    WRITELN('FALL/WINTER/SPRING/SUMMER');
    WRITE('******  *****  ******  **************           ');
    WRITELN('**** ****** ****** ******');
    WRITELN;
    FOR F := 1 TO NR_COURSES DO
      BEGIN  (* FOR F *)
        WITH DATA_FILE(.F.) DO
          BEGIN  (* WITH *)
            IF COURSENR(.1.) = '3' THEN
              BEGIN  (* IF 3 *)
                WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
              END;   (* IF 3 *)
          END;   (* WITH *)
      END;   (* FOR F *)
      WRITELN;  {*****************************************}
      WRITELN;  {*  THESE WRITELN'S ARE USED TO PUSH THE  *}
      WRITELN;  {*  LIST MENU TO THE NEXT FULL SCREEN     *}
      WRITELN;  {*****************************************}

END;   (* PROCEDURE *)

{*************************************************************************}
{* LIST 4000 LEVEL COURSES                                               *}
{*************************************************************************}

PROCEDURE LIST4;

VAR
   G           : INTEGER;   (* INDEX *)

BEGIN  (* PROCEDURE *)

  WRITELN('4000 LEVEL COURSES');
  WRITELN;                                                              ');
  WRITE('
  WRITELN('   ( 1 = YES / 0 = NO )');
  WRITE('COURSE  HOURS  CURRIC  PREREQUISITES            OFFERED: ');
  WRITELN('FALL/WINTER/SPRING/SUMMER');
  WRITE('******  *****  ******  **************           ');
```

```
    WRITELN('**** ****** ****** ******');
    WRITELN;
    FOR G := 1 TO NR_COURSES DO
      BEGIN  (* FOR G *)
        WITH DATA_FILE(.G.) DO
          BEGIN  (* WITH *)
            IF COURSENR(.1.) = '4' THEN
              BEGIN  (* IF 4 *)
                WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
              END;   (* IF 4 *)
          END;   (* WITH *)
      END;   (* FOR G *)
  END;   (* PROCEDURE *)

{*************************************************************************}
{* LIST ALL COURSES IN THE DATABASE                                      *}
{*************************************************************************}

  PROCEDURE LISTALL;

  VAR
    H           : INTEGER;  (* INDEX *)

  BEGIN  (* PROCEDURE *)

    WRITELN('DATABASE COURSE LISTING');
    WRITELN;
    WRITE('                                                          ');
    WRITELN('   ( 1 = YES / 0 = NO )');
    WRITE('COURSE  HOURS  CURRIC  PREREQUISITES           OFFERED: ');
    WRITELN('FALL/WINTER/SPRING/SUMMER');
    WRITE('******  *****  ******  **************                    ');
    WRITELN('**** ****** ****** ******');
    WRITELN;
    FOR H := 1 TO NR_COURSES DO
      BEGIN  (* FOR H *)
        WITH DATA_FILE(.H.) DO
          BEGIN  (* WITH *)
                WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
          END;   (* WITH *)
      END;   (* FOR H *)
      WRITELN;
```

```
    WRITELN;
END;    (* PROCEDURE *)

{*************************************************************************}
{* LIST DATA ON ANY SELECTED COURSE                                      *}
{*************************************************************************}

PROCEDURE LISTIND;

CONST
    FACTOR     = 64;

VAR
    J,K          : INTEGER;      (* INDEX *)
    ID           : COURSECODE;
    NR           : COURSENUMB;
    FOUND        : BOOLEAN;

BEGIN  (* PROCEDURE *)

  FOUND := FALSE;     (* INITIALIZE *)
  WRITELN;
  WRITELN;
  WRITELN;
  WRITE('             ');
  WRITELN('ENTER THE SIX CHARACTER CODE OF THE COURSE TO LOOK AT');
  READLN(INPT,ID,NR);
  BEGIN  (* CONVERT *)
    FOR K := 1 TO STRLEN(ID) DO
      IF ID(.K.) IN (.'a'..'z'.) THEN
        ID(.K.) := CHR(ORD(ID(.K.)) + FACTOR);
  END;   (* CONVERT *)
  WRITELN(ID:8,NR);     (* ECHO *)
  FOR J := 1 TO NR_COURSES DO
    BEGIN  (* FOR J *)
      WITH DATA_FILE(.J.) DO
        BEGIN  (* WITH *)
          IF ID = COURSEID THEN
            BEGIN  (* IF ID *)
              IF NR = COURSENR THEN
                BEGIN  (* IF NR *)
                  FOUND := TRUE;
                  WRITELN;
                  WRITE('                                ');
                  WRITE('                                ');
                  WRITELN('  ( 1 = YES / 0 = NO )');
```

```
                    WRITE('COURSE  HOURS  CURRIC  ');
                    WRITE('PREREQUISITES              OFFERED: ');
                    WRITELN('FALL/WINTER/SPRING/SUMMER');
                    WRITE('******  *****  ******  ');
                    WRITE('************                    ');
                    WRITELN('**** ****** ****** ******');
                    WRITELN;
                    WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                    WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                    WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                    WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
                    WRITELN;
                  END;    (* IF NR *)
              END;    (* IF ID *)
          END;    (* WITH *)
      END;    (* FOR J *)

      IF NOT FOUND THEN
        BEGIN
          WRITELN;
          WRITELN;
          WRITE('                              ');
          WRITELN(ID,NR,' IS NOT LOCATED IN THE DATA BASE');
          WRITELN;
          WRITELN;
          WRITELN;
          WRITELN;
        END;

      WRITELN; (**)
      WRITELN;  (**)
      WRITELN;   (*********************************************)
      WRITELN;    (*  THESE WRITELN'S ARE USED TO PUSH THE  *)
      WRITELN;    (*  LIST MENU TO THE NEXT FULL SCREEN     *)
      WRITELN;   (*********************************************)
      WRITELN;  (**)
      WRITELN; (**)

END;    (* PROCEDURE *)

(*****************************************************************)
(* LIST COURSES IN ANY EMPHASIS AREA                             *)
(*****************************************************************)

PROCEDURE LISTEMP;

VAR
   OK          : BOOLEAN;
```

```
    L,M          : INTEGER;       (* INDEX *)
    EMPANS       : CHAR;

BEGIN   (* PROCEDURE *)

    OK := FALSE;
    REPEAT      (* UNTIL OK *)
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN('    SELECT WHICH EMPHASIS AREA COURSES YOU WISH TO LOOK AT:');
    WRITELN;
    WRITELN('           ENTER:                                          ');
    WRITELN('             "C" FOR COMPUTER CENTER AND NETWORK OPERATIONS');
    WRITELN('             "I" FOR INFORMATION AND COMPUTER NETWORKS');
    WRITELN('             "T" FOR TACTICAL SYSTEMS');
    WRITELN('             "D" FOR DECISION SUPPORT SYSTEMS');
    WRITELN;
    READLN(INPT,EMPANS);
    WRITELN(EMPANS);           (* ECHO PRINT *)
    WRITELN;
       IF EMPANS IN (.'C','c','I','i','T','t','D','d'.) THEN
          ok := true
       else
          begin    (* else *)
             WRITELN('SORRY, WRONG INPUT, TRY AGAIN');
             PAGE;    (* CLEARS SCREEN *)
          end;     (* else *)
    UNTIL OK;  (* END REPEAT LOOP *)

    PAGE;  (* CLEAR SCREEN *)

    IF EMPANS IN (.'C','c'.) THEN
       begin  (* if c *)
          WRITE('YOU HAVE SELECTED "COMPUTER CENTER AND NETWORK ');
          WRITELN('OPERATIONS"');
          writeln;
          FOR L := 1 TO NR_COURSES DO
             BEGIN   (* FOR L *)
             WITH DATA_FILE(.L.) DO
                BEGIN   (* WITH *)
                IF CCNO = 2 THEN
                   BEGIN   (* IF 2 *)
                   WRITE('THE FOLLOWING COURSE IS REQUIRED FOR THIS');
                   WRITELN(' EMPHASIS AREA:');
                   WRITE('                                  ');
                   WRITE('                                        ');
```

```
                    WRITELN('   ( 1 = YES / 0 = NO )');
                    WRITE('COURSE  HOURS  CURRIC   ');
                    WRITE('PREREQUISITES          OFFERED: ');
                    WRITELN('FALL/WINTER/SPRING/SUMMER');
                    WRITE('******  *****  ******   ');
                    WRITE('************             ');
                    WRITELN('**** ****** ****** ******');
                    WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                    WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                    WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                    WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
                    WRITELN;
                    END;    (* IF 2 *)
                END;    (* WITH *)
            END;    (* FOR L *)
        WRITE('CHOOSE THREE (3) OF THE FOLLOWING COURSES AS ELECTIVES');
        WRITELN(' WITHIN THIS EMPHASIS AREA');
        WRITELN;
        WRITE('COURSE  HOURS  CURRIC   ');
        WRITE('PREREQUISITES          OFFERED: ');
        WRITELN('FALL/WINTER/SPRING/SUMMER');
        WRITE('******  *****  ******   ');
        WRITE('************             ');
        WRITELN('**** ****** ****** ******');
        FOR M := 1 TO NR_COURSES DO
            BEGIN   (* FOR M *)
            WITH DATA_FILE(.M.) DO
                BEGIN   (* WITH M *)
                IF CCNO = 1 THEN
                    BEGIN   (* IF 1 *)
                    WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                    WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                    WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                    WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
                    END;    (* IF 1 *)
                END;    (* WITH M *)
            END;    (* FOR M *)
    END     (* IF C *)

else
    IF EMPANS IN (.'I','i'.) THEN
        begin  (* if i *)
            WRITE('YOU HAVE SELECTED "INFORMATION AND COMPUTER ');
            WRITELN('NETWORKS"');
            writeln;
            FOR L := 1 TO NR_COURSES DO
                BEGIN   (* FOR L *)
                WITH DATA_FILE(.L.) DO
```

```
                BEGIN   (* WITH *)
                IF ICN = 2 THEN
                   BEGIN   (* IF 2 *)
                   WRITE('THE FOLLOWING COURSE IS REQUIRED FOR THIS');
                   WRITELN(' EMPHASIS AREA:');
                   WRITE('                              ');
                   WRITE('                                        ');
                   WRITELN('  ( 1 = YES / 0 = NO )');
                   WRITE('COURSE  HOURS  CURRIC  ');
                   WRITE('PREREQUISITES                OFFERED:');
                   WRITELN('FALL/WINTER/SPRING/SUMMER');
                   WRITE('******  *****  ******  ');
                   WRITE('************                  ');
                   WRITELN('**** ****** ****** ******');
                   WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                   WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                   WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                   WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
                   WRITELN;
                   END;    (* IF 2 *)
                END;    (* WITH *)
             END;    (* FOR L *)
          WRITELN;
          WRITE('CHOOSE THREE (3) OF THE FOLLOWING COURSES AS');
          WRITELN(' ELECTIVES WITHIN THIS EMPHASIS AREA');
          WRITELN;
          WRITE('COURSE  HOURS  CURRIC  ');
          WRITE('PREREQUISITES                OFFERED:');
          WRITELN('FALL/WINTER/SPRING/SUMMER');
          WRITE('******  *****  ******  ');
          WRITE('************                  ');
          WRITELN('**** ****** ****** ******');
          FOR M := 1 TO NR_COURSES DO
             BEGIN   (* FOR M *)
             WITH DATA_FILE(.M.) DO
                BEGIN   (* WITH M *)
                IF ICN = 1 THEN
                   BEGIN   (* IF 1 *)
                   WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                   WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                   WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                   WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
                   END;    (* IF 1 *)
                END;    (* WITH M *)
             END;    (* FOR M *)
       END    (* IF I *)
    else
       IF EMPANS IN (.'T','t'.) THEN
```

```
begin  (* if_t *)
   WRITELN('YOU HAVE SELECTED "TACTICAL SYSTEMS" ');
   writeln;
   FOR L := 1 TO NR_COURSES DO
      BEGIN   (* FOR L *)
      WITH DATA_FILE(.L.) DO
         BEGIN   (* WITH *)
         IF TS = 2 THEN
            BEGIN   (* IF 2 *)
            WRITE('THE FOLLOWING COURSE IS REQUIRED');
            WRITELN(' FOR THIS EMPHASIS AREA: ');
            WRITE('                                  ');
            WRITE('                                  ');
            WRITELN('   ( 1 = YES / 0 = NO )');
            WRITE('COURSE  HOURS  CURRIC   ');
            WRITE('PREREQUISITES                OFFERED: ');
            WRITELN('FALL/WINTER/SPRING/SUMMER');
            WRITE('******  *****  ******   ');
            WRITE('**************                    ');
            WRITELN('**** ****** ****** ******');
            WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
            WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
            WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
            WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
            WRITELN;
            END;    (* IF 2 *)
         END;    (* WITH *)
      END;    (* FOR L *)
   WRITE('CHOOSE THREE (3) OF THE FOLLOWING COURSES');
   WRITELN(' AS ELECTIVES WITHIN THIS EMPHASIS AREA');
   WRITELN;
   WRITE('COURSE  HOURS  CURRIC   ');
   WRITE('PREREQUISITES                OFFERED: ');
   WRITELN('FALL/WINTER/SPRING/SUMMER');
   WRITE('******  *****  ******   ');
   WRITE('**************                    ');
   WRITELN('**** ****** ****** ******');
   FOR M := 1 TO NR_COURSES DO
      BEGIN   (* FOR M *)
      WITH DATA_FILE(.M.) DO
         BEGIN   (* WITH M *)
         IF TS = 1 THEN
            BEGIN   (* IF 1 *)
            WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
            WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
            WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
            WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
            END;    (* IF 1 *)
```

```
                END;     (* WITH M *)
            END;     (* FOR M *)
    END     (* IF T *)

else
    IF EMPANS IN ['D','d'] THEN
    begin  (* if d *)
        WRITE('YOU HAVE SELECTED "DECISION SUPPORT ');
        WRITELN('SYSTEMS"');
        writeln;
        FOR L := 1 TO NR_COURSES DO
            BEGIN   (* FOR L *)
            WITH DATA_FILE[L] DO
                BEGIN   (* WITH *)
                IF DSS = 2 THEN
                    BEGIN   (* IF 2 *)
                    WRITE('THE FOLLOWING COURSE IS REQUIRED');
                    WRITELN(' FOR THIS EMPHASIS AREA:');
                    WRITE('                                  ');
                    WRITE('                                  ');
                    WRITELN('  ( 1 = YES / 0 = NO )');
                    WRITE('COURSE  HOURS  CURRIC   ');
                    WRITE('PREREQUISITES                OFFERED:');
                    WRITELN('FALL/WINTER/SPRING/SUMMER');
                    WRITE('******  *****  ******   ');
                    WRITE('************                  ');
                    WRITELN('**** ****** ****** ******');
                    WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                    WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                    WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                    WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
                    WRITELN;
                    END;    (* IF 2 *)
                END;    (* WITH *)
            END;    (* FOR L *)
        WRITE('CHOOSE THREE (3) OF THE FOLLOWING COURSES');
        WRITELN(' AS ELECTIVES WITHIN THIS EMPHASIS AREA');
        WRITELN;
        WRITE('COURSE  HOURS  CURRIC   ');
        WRITE('PREREQUISITES                OFFERED:');
        WRITELN('FALL/WINTER/SPRING/SUMMER');
        WRITE('******  *****  ******   ');
        WRITE('************                  ');
        WRITELN('**** ****** ****** ******');
        FOR M := 1 TO NR_COURSES DO
            BEGIN   (* FOR M *)
            WITH DATA_FILE[M] DO
                BEGIN   (* WITH M *)
```

```
                        IF DSS = 1 THEN
                           BEGIN   (* IF 1 *)
                           WRITE(COURSEID,COURSENR,HOURS:5,CURRIC:8);
                           WRITE(PREID1:6,PRENR1,PREID2:4,PRENR2);
                           WRITE(PREID3:4,PRENR3,PREID4:4,PRENR4);
                           WRITELN(FALL:5,WINTER:6,SPRING:7,SUMMER:7);
                           END;    (* IF 1 *)
                        END;    (* WITH M *)
                     END;    (* FOR M *)
                end;   (* if D *)

END;    (* PROCEDURE *)

{*****************************************************************}
{* LIST COURSE NAMES UPON REQUEST                                *}
{*****************************************************************}

PROCEDURE NAMES;

VAR
   U          : INTEGER;  (* INDEX *)

BEGIN (* PROCEDURE *)

   WRITE('  COURSE   COURSE NAME                    ');
   WRITELN('                (LECTURE/LAB HOURS)');
   WRITE('  ******   **********                    ');
   WRITELN('                ********************');
   WRITELN;
   FOR U := 1 TO NR_COURSES DO
      BEGIN (* FOR U *)
         WITH DATA_FILE(.U.) DO
            BEGIN (* WITH *)
              WRITELN('  ',DESCRIPT(.1.));
            END;  (* WITH *)
      END;  (* FOR U *)
      WRITELN; (**)
      WRITELN;  {*****************************************************}
      WRITELN;  {* WRITELN'S TO PUSH LIST MENU TO NEXT FULL SCREEN   *}
      WRITELN;  {*****************************************************}
      WRITELN; (**)

END;  (* PROCEDURE *)

{*****************************************************************}
{* LIST DESCRIPTIONS ON SELECTED COURSES                         *}
{*****************************************************************}
```

```
PROCEDURE DESCRIBE;

CONST
    FACTOR     = 64;

VAR

   BB,CC,K     : INTEGER;      (* INDEX *)
   ID          : COURSECODE;
   NR          : COURSENUMB;
   FOUND,DONE  : BOOLEAN;
   ANSWER      : CHAR;

BEGIN  (* PROCEDURE *)

  DONE := FALSE;     (* INITIALIZE *)
  REPEAT  (* UNTIL DONE *)
  FOUND := FALSE;    (* INITIALIZE *)
  WRITELN;
  WRITELN;
  WRITELN;
  WRITE('                ');
  WRITELN('ENTER THE SIX CHARACTER CODE OF THE COURSE TO LOOK AT');
  READLN(INPT,ID,NR);
  BEGIN  (* CONVERT *)
    FOR K := 1 TO STRLEN(ID) DO
     IF ID(.K.) IN ('a'..'z') THEN
       ID(.K.) := CHR(ORD(ID(.K.)) + FACTOR);
  END;   (* CONVERT *)
  WRITELN(ID,NR);   (* ECHO *)
  PAGE;             (* CLEAR SCREEN *)
  FOR BB := 1 TO NR_COURSES DO
    BEGIN  (* FOR BB *)
      WITH DATA_FILE(.BB.) DO
        BEGIN  (* WITH *)
          IF ID = COURSEID THEN
            BEGIN  (* IF ID *)
              IF NR = COURSENR THEN
                BEGIN  (* IF NR *)
                  FOUND := TRUE;
 (* HEADER *)     WRITELN;
                  WRITELN;
                  WRITELN;
                  WRITELN;
                  WRITELN;
                  FOR CC := 1 TO 10 DO
                    BEGIN  (* FOR CC *)
                      WRITELN(DESCRIPT(.CC.));
```

```
                        END;    (* FOR CC *)
(* TRAILER *)    WRITELN;
                 WRITELN;
                 WRITELN;
                 WRITELN;
                 WRITELN;
                 WRITELN;
               END;    (* IF NR *)
          END;    (* IF ID *)
       END;    (* WITH *)
  END;    (* FOR BB *)

  IF NOT FOUND THEN
    BEGIN
       WRITELN;
       WRITELN;
       WRITE('                              ');
       WRITELN(ID,NR,' IS NOT LOCATED IN THE DATA BASE');
       WRITELN;
       WRITELN;
       WRITELN;
       WRITELN;
    END;

    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN;
    WRITE('                              ');
    WRITELN('DO YOU WISH TO SEE ANY OTHER COURSE DESCRIPTIONS?');
    WRITE('                              ');
    WRITELN('ENTER Y FOR YES / ANY OTHER CHARACTER FOR NO');
    WRITELN;
    READLN(INPT,ANSWER);
    WRITELN(ANSWER);              (* ECHO PRINT *)

    IF ANSWER IN ['Y','y'] THEN
     BEGIN  (* IF Y *)
       DONE := FALSE;
       PAGE;  (* CLEAR SCREEN *)
     END    (* IF Y *)
    ELSE
     BEGIN (* ELSE *)
       DONE := TRUE;
       PAGE;   (* CLEAR SCREEN *)
     END;  (* ELSE *)

UNTIL DONE;  (* END REPEAT LOOP *)
```

```
  PAGE;         (* CLEAR SCREEN *)
END;  (* PROCEDURE *)

{*************************************************************************}
{* PROCEDURE TO CALL OTHER PROCEDURES TO LIST COURSES AS REQUIRED        *}
{*************************************************************************}

PROCEDURE LISTER;

VAR
   GOOD,MOVE  : BOOLEAN;
   LSTANS     : CHAR;

BEGIN  (* PROCEDURE *)

  MOVE := FALSE;              (* INITIALIZE *)
  REPEAT  (* UNTIL MOVE *)
    GOOD := FALSE;            (* INITIALIZE *)
    REPEAT  (* UNTIL GOOD *)
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN('              ***************** LIST MENU ******************');
    WRITELN('              *                                            *');
    WRITELN('              *  ENTER:                                    *');
    WRITELN('              *    "Q" TO QUIT LISTING                     *');
    WRITELN('              *    "A" TO SEE ALL ADMIN SCIENCE COURSES    *');
    WRITELN('              *    "C" TO SEE ALL COMPUTER SCIENCE COURSES *');
    WRITELN('              *    "3" TO SEE ALL 3000 LEVEL COURSES       *');
    WRITELN('              *    "4" TO SEE ALL 4000 LEVEL COURSES       *');
    WRITELN('              *    "E" TO SEE ALL EMPHASIS AREA COURSES    *');
    WRITELN('              *    "R" TO SEE ALL REQUIRED COURSES         *');
    WRITELN('              *    "I" TO SEE DATA ON AN INDIVIDUAL COURSE *');
    WRITELN('              *    "B" TO SEE ALL COURSES IN THE DATA BASE *');
    WRITELN('              *    "N" TO SEE ALL COURSE "NAMES/TITLES"    *');
    WRITELN('              *    "D" TO SEE COURSE DESCRIPTIONS          *');
    WRITELN('              *                                            *');
    WRITELN('              ***************** LIST MENU ******************');
    READLN(INPT,LSTANS);
    WRITELN(LSTANS);      (* ECHO *)
      IF LSTANS IN ['Q','q','A','a','C','c','3','4','E','e'] THEN
         GOOD := TRUE
      ELSE
      IF LSTANS IN ['R','r','I','i','B','b','N','n','D','d'] THEN
         GOOD := TRUE
      ELSE
       BEGIN (* ELSE *)
```

```
        PAGE;  (* CLEAR SCREEN *)
      END;  (* ELSE *)

    UNTIL GOOD;  (* END REPEAT LOOP *)

    PAGE;  (* CLEAR SCREEN *)

    IF LSTANS IN (.'Q','q'.) THEN
       MOVE := TRUE
    ELSE
    IF LSTANS IN (.'A','a'.) THEN
       LISTAS
    ELSE
    IF LSTANS IN (.'C','c'.) THEN
       LISTCS
    ELSE
    IF LSTANS = '3' THEN
       LIST3
    ELSE
    IF LSTANS = '4' THEN
       LIST4
    ELSE
    IF LSTANS IN (.'E','e'.) THEN
       LISTEMP
    ELSE
    IF LSTANS IN (.'R','r'.) THEN
       LIST_REQUIRED_COURSES
    ELSE
    IF LSTANS IN (.'I','i'.) THEN
       LISTIND
    ELSE
    IF LSTANS IN (.'B','b'.) THEN
       LISTALL
    ELSE
    IF LSTANS IN (.'N','n'.) THEN
       NAMES
    ELSE
    IF LSTANS IN (.'D','d'.) THEN
       DESCRIBE;

  UNTIL MOVE;  (* END REPEAT LOOP *)

  PAGE;  (* CLEAR SCREEN *)

END;   (* PROCEDURE *)

(*****************************************************************)
(* CHECK IF ALL PREREQUISITES HAVE BEEN MEET FOR SCHEDULED COURSES   *)
```

```
(*************************************************************************)

PROCEDURE PREREQUISITE_CHECK;

VAR
   P,Q,U,V,W  : INTEGER;  (* INDEX *)
   CHECK      : BOOLEAN;

BEGIN  (* PROCEDURE *)

  CHECK := TRUE;
  FOR P := 1 TO NR_COURSES DO
    BEGIN  (* FOR P *)
    IF DATA_FILE(.P.).TERM > 1 THEN    (* ON THE SCHEDULE *)
      BEGIN  (* IF P-TERM > 1 *)
      IF DATA_FILE(.P.).PREID1 <> '   ' THEN
        BEGIN  (* PREID1 NOT BLANK *)
        FOR Q := 1 TO NR_COURSES DO
          BEGIN  (* FOR Q *)
          IF DATA_FILE(.P.).PREID1 = DATA_FILE(.Q.).COURSEID THEN
            BEGIN  (* IF ID *)
            IF DATA_FILE(.P.).PRENR1 = DATA_FILE(.Q.).COURSENR THEN
              BEGIN  (* IF NR *)
              IF DATA_FILE(.Q.).TERM = 0 THEN  (* PREREQ NOT SCHED *)
                BEGIN  (* Q-TERM = 0 *)
                  CHECK := FALSE;
                  WRITELN;
                  WRITE(DATA_FILE(.Q.).COURSEID,DATA_FILE(.Q.).COURSENR);
                  WRITE(' IS A PREREQUISITE FOR ');
                  WRITE(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
                  WRITELN(' AND HAS NOT BEEN SCHEDULED');
                  WRITE('THIS PREREQUISITE MUST BE ');
                  WRITE('SCHEDULED PRIOR TO TERM ');
                  WRITELN(DATA_FILE(.P.).TERM - 1:2);
                END    (* Q-TERM = 0 *)
              ELSE
              IF DATA_FILE(.P.).TERM <= DATA_FILE(.Q.).TERM THEN
                (* PREREQ IS SCHED TOO LATE OR CONCURRENTLY *)
                BEGIN  (* IF P-TERM <= Q-TERM *)
                CHECK := FALSE;
                WRITELN;
                WRITE(DATA_FILE(.Q.).COURSEID,DATA_FILE(.Q.).COURSENR);
                WRITE(' IS A PREREQUISITE FOR ');
                WRITE(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
                WRITE(' & IS SCHEDULED AFTER/CONCURRENT WITH ');
                WRITELN(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
                WRITE('THIS PREREQUISITE MUST BE ');
                WRITE('SCHEDULED PRIOR TO TERM ');
```

```
            WRITELN(DATA_FILE(.P.).TERM - 1:2);
            END;    (* IF P-TERM <= Q-TERM *)
          END;    (* IF NR *)
        END;    (* IF ID *)
      END;  (* FOR Q *)

IF DATA_FILE(.P.).PREID2 <> '    ' THEN
  BEGIN  (* PREID2 NOT BLANK *)
  FOR U := 1 TO NR_COURSES DO
    BEGIN  (* FOR U *)
    IF DATA_FILE(.P.).PREID2 = DATA_FILE(.U.).COURSEID THEN
      BEGIN  (* IF ID *)
      IF DATA_FILE(.P.).PRENR2 = DATA_FILE(.U.).COURSENR THEN
        BEGIN  (* IF NR *)
        IF DATA_FILE(.U.).TERM = 0 THEN  (* PREREQ NOT SCHED *)
          BEGIN  (* U-TERM = 0 *)
            CHECK := FALSE;
            WRITELN;
            WRITE(DATA_FILE(.U.).COURSEID,DATA_FILE(.U.).COURSENR);
            WRITE('  IS A PREREQUISITE FOR ');
            WRITE(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
            WRITELN('  AND HAS NOT BEEN SCHEDULED');
            WRITE('THIS PREREQUISITE MUST BE ');
            WRITE('SCHEDULED PRIOR TO TERM ');
            WRITELN(DATA_FILE(.P.).TERM - 1:2);
          END    (* U-TERM = 0 *)
        ELSE
        IF DATA_FILE(.P.).TERM <= DATA_FILE(.U.).TERM THEN
          (* PREREQ IS SCHED TOO LATE OR CONCURRENTLY *)
          BEGIN  (* IF P-TERM <= U-TERM *)
          CHECK := FALSE;
          WRITELN;
          WRITE(DATA_FILE(.U.).COURSEID,DATA_FILE(.U.).COURSENR);
          WRITE('  IS A PREREQUISITE FOR ');
          WRITE(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
          WRITE('  & IS SCHEDULED AFTER/CONCURRENT WITH ');
          WRITELN(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
          WRITE('THIS PREREQUISITE MUST BE ');
          WRITE('SCHEDULED PRIOR TO TERM ');
          WRITELN(DATA_FILE(.P.).TERM - 1:2);
          END;    (* IF P-TERM <= U-TERM *)
        END;    (* IF NR *)
      END;    (* IF ID *)
    END;  (* FOR U *)

IF DATA_FILE(.P.).PREID3 <> '    ' THEN
  BEGIN  (* PREID3 NOT BLANK *)
  FOR V := 1 TO NR_COURSES DO
```

```
      BEGIN  (* FOR V *)
      IF DATA_FILE(.P.).PREID3 = DATA_FILE(.V.).COURSEID THEN
        BEGIN  (* IF ID *)
        IF DATA_FILE(.P.).PRENR3 = DATA_FILE(.V.).COURSENR THEN
          BEGIN  (* IF NR *)
          IF DATA_FILE(.V.).TERM = 0 THEN  (* PREREQ NOT SCHED *)
            BEGIN  (* V-TERM = 0 *)
              CHECK := FALSE;
              WRITELN;
              WRITE(DATA_FILE(.V.).COURSEID,DATA_FILE(.V.).COURSENR);
              WRITE(' IS A PREREQUISITE FOR ');
              WRITE(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
              WRITELN(' AND HAS NOT BEEN SCHEDULED');
              WRITE('THIS PREREQUISITE MUST BE ');
              WRITE('SCHEDULED PRIOR TO TERM ');
              WRITELN(DATA_FILE(.P.).TERM - 1:2);
            END    (* V-TERM = 0 *)
          ELSE
          IF DATA_FILE(.P.).TERM <= DATA_FILE(.V.).TERM THEN
            (* PREREQ IS SCHED TOO LATE OR CONCURRENTLY *)
            BEGIN  (* IF P-TERM <= V-TERM *)
            CHECK := FALSE;
            WRITELN;
            WRITE(DATA_FILE(.V.).COURSEID,DATA_FILE(.V.).COURSENR);
            WRITE(' IS A PREREQUISITE FOR ');
            WRITE(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
            WRITE(' & IS SCHEDULED AFTER/CONCURRENT WITH ');
            WRITELN(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
            WRITE('THIS PREREQUISITE MUST BE ');
            WRITE('SCHEDULED PRIOR TO TERM ');
            WRITELN(DATA_FILE(.P.).TERM - 1:2);
            END;   (* IF P-TERM <= V-TERM *)
          END;   (* IF NR *)
        END;   (* IF ID *)
      END;  (* FOR V *)

    IF DATA_FILE(.P.).PREID4 <> '    ' THEN
      BEGIN  (* PREID4 NOT BLANK *)
      FOR W := 1 TO NR_COURSES DO
        BEGIN  (* FOR W *)
        IF DATA_FILE(.P.).PREID4 = DATA_FILE(.W.).COURSEID THEN
          BEGIN  (* IF ID *)
          IF DATA_FILE(.P.).PRENR4 = DATA_FILE(.W.).COURSENR THEN
            BEGIN  (* IF NR *)
            IF DATA_FILE(.W.).TERM = 0 THEN  (* PREREQ NOT SCHED *)
              BEGIN  (* W-TERM = 0 *)
                CHECK := FALSE;
                WRITELN;
```

```
                        WRITE(DATA_FILE(.W.).COURSEID,DATA_FILE(.W.).COURSENR);
                        WRITE('  IS A PREREQUISITE FOR  ');
                        WRITE(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
                        WRITELN('  AND HAS NOT BEEN SCHEDULED');
                        WRITE('THIS PREREQUISITE MUST BE  ');
                        WRITE('SCHEDULED PRIOR TO TERM  ');
                        WRITELN(DATA_FILE(.P.).TERM - 1:2);
                      END     (* W-TERM = 0 *)
                    ELSE
                    IF DATA_FILE(.P.).TERM <= DATA_FILE(.W.).TERM THEN
                      (* PREREQ IS SCHED TOO LATE OR CONCURRENTLY *)
                      BEGIN  (* IF P-TERM <= W-TERM *)
                      CHECK := FALSE;
                      WRITELN;
                      WRITE(DATA_FILE(.W.).COURSEID,DATA_FILE(.W.).COURSENR);
                      WRITE('  IS A PREREQUISITE FOR  ');
                      WRITE(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
                      WRITE('  & IS SCHEDULED AFTER/CONCURRENT WITH  ');
                      WRITELN(DATA_FILE(.P.).COURSEID,DATA_FILE(.P.).COURSENR);
                      WRITE('THIS PREREQUISITE MUST BE  ');
                      WRITE('SCHEDULED PRIOR TO TERM  ');
                      WRITELN(DATA_FILE(.P.).TERM - 1:2);
                      END;    (* IF P-TERM <= W-TERM *)
                  END;    (* IF NR *)
                END;    (* IF ID *)
              END;  (* FOR W *)

          END;    (* PREID4 NOT BLANK *)
          END;    (* PREID3 NOT BLANK *)
          END;    (* PREID2 NOT BLANK *)
          END;    (* PREID1 NOT BLANK *)
        END;    (* IF P-TERM > 1 *)
      END;    (* FOR P *)
    IF CHECK THEN
     BEGIN
      WRITELN;
      WRITELN('    ALL PREREQUISITES HAVE BEEN SCHEDULED APPROPRIATELY');
     END;

END;    (* PROCEUDRE *)

{*****************************************************************}
{* ADD A COURSE TO THE SCHEDULE                                  *}
{*****************************************************************}

PROCEDURE ADD_A_COURSE;

CONST
```

```
    FACTOR     = 64;

VAR
   B,K          : INTEGER;      (* INDEX *)
   ID           : COURSECODE;
   NR           : COURSENUMB;
   FOUND,DONE   : BOOLEAN;
   OK_TERM      : BOOLEAN;
   OFFERED      : BOOLEAN;
   ANSWER       : CHAR;
   TERMNR       : INTEGER;
   STAY,ADD     : BOOLEAN;
   ADDANS       : CHAR;
   STILANS      : CHAR;

{**************************************************************************}
{* IF COURSE TO ADD NOT OFFERED IN THE DESIRED TERM, CHOOSE AGAIN?        *}
{*              INTERNAL TO PROCEDURE ADD_A_COURSE                        *}
{**************************************************************************}

PROCEDURE PICK_A_NEW_TERM (VAR FINISHED: BOOLEAN; VAR OK : BOOLEAN);

VAR
   AGAIN,STILL: CHAR;

BEGIN  (* PROCEDURE *)

  FINISHED := FALSE;  (* INITIALIZE *)
  OK := FALSE;        (* INITIALIZE *)
  WRITELN;
  WRITE('                    ');
  WRITELN('DO YOU WISH TO SCHEDULE IT IN ANOTHER TERM?');
  WRITE('                    ');
  WRITELN('ENTER "Y" FOR YES, ANY OTHER CHARACTER FOR NO');
  WRITELN;
  READLN(INPT,AGAIN);
  WRITELN(AGAIN);             (* ECHO PRINT *)
  IF AGAIN IN ['Y','y'] THEN
     FINISHED := FALSE    (* NOT FINISHED YET *)
  ELSE
     BEGIN  (* ELSE *)
        PAGE;             (* CLEAR SCREEN *)
        WRITELN;
        WRITELN;
        WRITELN;
        WRITE('                    ');
        WRITELN('IF YOU HAVE LEARNED THAT THE DATABASE IS NOT CORRECT');
        WRITE('                    ');
```

```
            WRITELN('AND THAT  THIS  COURSE WILL BE  OFFERED AS  PER YOUR ');
            WRITE('                    ');
            WRITELN('REQUEST, YOU MAY SCHEDULE IT ACCORDINGLY.');
            WRITELN;
            WRITE('                    ');
            WRITELN('IF THIS IS THE CASE, AND YOU WANT TO SCHEDULE');
            WRITE('                    ');
            WRITELN('THE COURSE AS ORIGINALLY REQUESTED,ENTER "Y".');
            WRITE('                    ');
            WRITELN('ENTER ANY OTHER CHARACTER FOR NO.');
            WRITELN;
            READLN(INPT,STILL);
            WRITELN(STILL);      (* ECHO *)
            IF STILL IN ['Y','y'] THEN
               OK := TRUE;
            FINISHED := TRUE;
         END;    (* ELSE *)

      END;    (* PROCEDURE *)
      (************************* INTERNAL PROCEDURE ************************)

      BEGIN  (* PROCEDURE *)

        STAY := TRUE;
        REPEAT    (* UNTIL NOT STAY *)

        ADD := FALSE;
        FOUND := FALSE;      (* INITIALIZE *)
        WRITELN;
        WRITELN;
        WRITELN;
        WRITELN;
        WRITELN('         ENTER THE SIX CHARACTER CODE OF THE COURSE TO ADD');
        READLN(INPT,ID,NR);
        BEGIN  (* CONVERT *)
          FOR K := 1 TO STRLEN(ID) DO
           IF ID[.K.] IN ['a'..'z'] THEN
             ID[.K.] := CHR(ORD(ID[.K.]) + FACTOR);
        END;    (* CONVERT *)
        WRITELN(ID:8,NR);   (* ECHO PRINT *)
        FOR B := 1 TO NR_COURSES DO
          BEGIN (* FOR B *)
            WITH DATA_FILE[.B.] DO
              BEGIN (* WITH *)
              IF ID = COURSEID THEN
                BEGIN (* IF ID *)
                IF NR = COURSENR THEN
                  BEGIN (* NR *)
```

```
FOUND := TRUE;
IF TERM = 1 THEN
  BEGIN (* IF = 1 *)
    WRITELN(ID,NR,' HAS BEEN VALIDATED.');
    WRITE('ADDING THIS COURSE WILL REMOVE IT FROM ');
    WRITELN('THE VALIDATED COURSE LIST.');
    WRITELN('DO YOU STILL WANT TO ADD IT? (Y/N)');
    READLN(INPT,STILANS);
    WRITELN(STILANS);   (* ECHO *)
    IF STILANS IN ['Y','y'] THEN
      BEGIN
       ADD := TRUE;
       PAGE;              (* CLEAR SCREEN *)
      END;
  END   (* IF = 1 *)
ELSE
IF TERM > 1 THEN
  BEGIN (* IF > 1 *)
    WRITELN;
    WRITE('                    ');
    WRITELN(ID,NR,' IS ALREADY SCHEDULED IN TERM ',TERM-1:2);
    WRITE('      ');
    WRITE('ADDING THIS COURSE WILL REMOVE IT FROM ');
    WRITELN('THE CURRENT SCHEDULE LOCATION.');
    WRITE('                    ');
    WRITELN('DO YOU STILL WANT TO ADD IT? (Y/N)');
    READLN(INPT,STILANS);
    WRITELN(STILANS);   (* ECHO *)
    IF STILANS IN ['Y','y'] THEN
      BEGIN
       ADD := TRUE;
       PAGE;              (* CLEAR SCREEN *)
      END;
  END   (* IF > 1 *)
ELSE
  BEGIN (* ELSE *)
     ADD := TRUE;
  END;  (* ELSE *)

IF ADD THEN
  BEGIN  (* IF ADD *)
  WRITELN;
  WRITELN;
  WRITELN;
  WRITELN;
  WRITE(ID,NR,' IS OFFERED IN THE ');
  IF FALL = 1 THEN
    BEGIN (* FALL *)
```

```
    WRITE ('/FALL ');
      IF TERM1 = 'F' THEN
        WRITE('(1,&,5)/');
      IF TERM2 = 'F' THEN
        WRITE('(2,&,6)/');
      IF TERM3 = 'F' THEN
        WRITE('(3)/');
      IF TERM4 = 'F' THEN
        WRITE('(4)/');
    END;  (* FALL *)
  IF WINTER = 1 THEN
    BEGIN (* WINTER *)
    WRITE ('/WINTER ');
      IF TERM1 = 'W' THEN
        WRITE('(1,&,5)/');
      IF TERM2 = 'W' THEN
        WRITE('(2,&,6)/');
      IF TERM3 = 'W' THEN
        WRITE('(3)/');
      IF TERM4 = 'W' THEN
        WRITE('(4)/');
    END;  (* WINTER *)
  IF SPRING = 1 THEN
    BEGIN (* SPRING *)
    WRITE ('/SPRING ');
      IF TERM1 = 'S' THEN
        WRITE('(1,&,5)/');
      IF TERM2 = 'S' THEN
        WRITE('(2,&,6)/');
      IF TERM3 = 'S' THEN
        WRITE('(3)/');
      IF TERM4 = 'S' THEN
        WRITE('(4)/');
    END;  (* SPRING *)
  IF SUMMER = 1 THEN
    BEGIN (* SUMMER *)
    WRITE ('/SUMMER ');
      IF TERM1 = 'R' THEN
        WRITE('(1,&,5)/');
      IF TERM2 = 'R' THEN
        WRITE('(2,&,6)/');
      IF TERM3 = 'R' THEN
        WRITE('(3)/');
      IF TERM4 = 'R' THEN
        WRITE('(4)/');
    END;  (* SUMMER *)
  WRITELN;
```

```
DONE := FALSE;                       (* INITIALIZE *)
REPEAT  (* UNTIL DONE *)
  OK_TERM := FALSE;                  (* INITIALIZE *)
  REPEAT  (* UNTIL OK_TERM *)
    WRITELN;
    WRITE('        WHAT TERM (#) DO YOU WISH TO SCHEDULE ');
    WRITELN( ID,NR,' IN?');
    WRITELN;
    READLN( INPT,ANSWER);
    WRITELN( ANSWER);          (* ECHO PRINT *)
    WRITELN;
    IF ANSWER IN (.'1','2','3','4','5','6'.) THEN
       OK_TERM := TRUE
    ELSE
      BEGIN
        PAGE;  (* CLEAR SCREEN *)
        WRITELN;
        WRITELN;
        WRITELN;
        WRITELN;
        WRITE('                 ');
        WRITELN('SORRY, INPUT MUST BE 1,2,3,4,5 OR 6');
      END;
  UNTIL OK_TERM;    (* END REPEAT LOOP *)
  PAGE;  (* CLEAR SCREEN *)
  BEGIN (* CORROLATE TERM TO QTR *)
    IF ANSWER = '1' THEN          (**)
      BEGIN  (* IF 1 *)           (**)
        TERMNR := 2;              (**)
        TERMCHOICE := TERM1;      (**)
      END    (* IF 1 *)           (**)
    ELSE                          (**)
    IF ANSWER = '2' THEN          (**)
      BEGIN  (* IF 2 *)           (**)
        TERMNR := 3;              (**)
        TERMCHOICE := TERM2;      (**)
      END    (* IF 2 *)           (**)
    ELSE                          (**)
    IF ANSWER = '3' THEN          (**)
      BEGIN  (* IF 3 *)           (**)
        TERMNR := 4;              (**)
        TERMCHOICE := TERM3;      (**)
      END    (* IF 3 *)           (*****************)
    ELSE                          (*    CORROLATE   *)
    IF ANSWER = '4' THEN          (* TERM TO QUARTER *)
      BEGIN  (* IF 4 *)           (*****************)
        TERMNR := 5;              (**)
        TERMCHOICE := TERM4;      (**)
```

```
    END    (* IF 4 *)                    {**}
  ELSE                                    {**}
  IF ANSWER = '5' THEN                    {**}
    BEGIN  (* IF 5 *)                     {**}
      TERMNR := 6;                        {**}
      TERMCHOICE := TERM5;                {**}
    END    (* IF 5 *)                    {**}
  ELSE                                    {**}
  IF ANSWER = '6' THEN                    {**}
    BEGIN  (* IF 6 *)                     {**}
      TERMNR := 7;                        {**}
      TERMCHOICE := TERM6;                {**}
    END;   (* IF 6 *)                    {**}
END;  (* CORROLATE TERM TO QTR *)

OFFERED := FALSE;
IF TERMCHOICE = 'F' THEN
   BEGIN (* IF F *)
   IF FALL = 1 THEN
      BEGIN (* IF FALL *)
      OFFERED := TRUE;
      DONE := TRUE;
      END   (* IF FALL *)
   ELSE
      BEGIN (* ELSE *)
        WRITELN;
        WRITELN;
        WRITE('                    ');
        WRITE('SORRY, ',ID,NR,' IS NOT OFFERED IN THE ');
        WRITELN('FALL');
        PICK_A_NEW_TERM(DONE,OFFERED);(* PROCEDURE *)
        PAGE;  (* CLEAR SCREEN *)
      END;  (* ELSE *)
   END   (* IF F *)
ELSE
IF TERMCHOICE = 'W' THEN
   BEGIN (* IF W *)
   IF WINTER = 1 THEN
      BEGIN  (* IF WINTER *)
      OFFERED := TRUE;
      DONE := TRUE;
      END    (* IF WINTER *)
   ELSE
      BEGIN (* ELSE *)
        WRITELN;
        WRITELN;
        WRITE('                    ');
        WRITE('SORRY, ',ID,NR,' IS NOT OFFERED IN THE ');
```

```
            WRITELN('WINTER');
            PICK_A_NEW_TERM(DONE,OFFERED);  (* PROCEDURE *)
            PAGE;  (* CLEAR SCREEN *)
          END;  (* ELSE *)
      END   (* IF W *)
  ELSE
  IF TERMCHOICE = 'S' THEN
      BEGIN (* IF S *)
      IF SPRING = 1 THEN
          BEGIN (* IF SPRING *)
          OFFERED := TRUE;
          DONE := TRUE;
          END   (* IF SPRING *)
      ELSE
          BEGIN (* ELSE *)
            WRITELN;
            WRITELN;
            WRITE('                    ');
            WRITE('SORRY, ',ID,NR,' IS NOT OFFERED IN THE ');
            WRITELN('SPRING');
            PICK_A_NEW_TERM(DONE,OFFERED);  (* PROCEDURE *)
            PAGE;  (* CLEAR SCREEN *)
          END;  (* ELSE *)
      END   (* IF S *)
  ELSE
  IF TERMCHOICE = 'R' THEN
      BEGIN (* IF R *)
      IF SUMMER = 1 THEN
          BEGIN (* IF SUMMER *)
          OFFERED := TRUE;
          DONE := TRUE;
          END   (* IF SUMMER *)
      ELSE
          BEGIN (* ELSE *)
            WRITELN;
            WRITELN;
            WRITE('                    ');
            WRITE('SORRY, ',ID,NR,' IS NOT OFFERED IN THE ');
            WRITELN('SUMMER');
            PICK_A_NEW_TERM(DONE,OFFERED);  (* PROCEDURE *)
            PAGE;  (* CLEAR SCREEN *)
          END;  (* ELSE *)
      END;  (* IF R *)
UNTIL DONE;   (* END OF REPEAT LOOP *)

IF OFFERED THEN    (* OFFERED IN REQUESTED TERM *)
  BEGIN (* IF OFFERED *)
    TERM := TERMNR;
```

```
                        WRITELN;
                        WRITELN;
                        WRITELN;
                        WRITELN;
                        WRITE('                    ');
                        WRITELN(ID,NR,' HAS BEEN ADDED TO YOUR SCHEDULE');
                      END;  (* IF OFFERED *)
                    END;  (* IF ADD *)
                  END;  (* IF NR *)
                END;  (* IF ID *)

            END;  (* WITH *)
        END;  (* FOR B *)

        IF NOT FOUND THEN
         BEGIN
           WRITELN;
           WRITE('                    ');
           WRITELN(ID,NR,' IS NOT LOCATED IN THE DATABASE');
         END;

      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN;
      WRITELN('                    DO YOU WANT TO ADD ANOTHER COURSE?');
      WRITELN('                    ENTER "Y" FOR YES, ANY OTHER FOR NO');
      READLN(INPT,ADDANS);
      WRITELN(ADDANS);      (* ECHO *)
      IF ADDANS IN ['Y','y'] THEN
         BEGIN  (* IF Y *)
            STAY := TRUE;
            PAGE;           (* CLEAR SCREEN *)
         END    (* IF Y *)
      ELSE
            STAY := FALSE;
      UNTIL NOT STAY;       (* END REPEAT LOOP *)
      PAGE;  (* CLEAR SCREEN *)

END;    (* PROCEDURE *)

{*****************************************************************}
{*                    CHOOSE AN EMPHASIS AREA                    *}
{*****************************************************************}

PROCEDURE EMPHASIS_AREA;

VAR
```

```
   OK          : BOOLEAN;
   LOOK,NOW    : CHAR;

BEGIN   (* PROCEDURE *)

 OK := FALSE;
 REPEAT      (* UNTIL OK *)
 WRITELN;
 WRITELN;
 WRITELN;
 WRITELN;
 WRITELN('            ************ SELECT EMPHASIS MENU ***************');
 WRITELN('            *                                               *');
 WRITELN('            *  SELECT AN EMPHASIS AREA:                     *');
 WRITELN('            *                                               *');
 WRITELN('            *  ENTER "C" COMPUTER CENTER AND                *');
 WRITELN('            *                          NETWORK OPERATIONS   *');
 WRITELN('            *  ENTER "I" INFORMATION AND COMPUTER NETWORKS  *');
 WRITELN('            *  ENTER "T" TACTICAL SYSTEMS                   *');
 WRITELN('            *  ENTER "D" DECISION SUPPORT SYSTEMS           *');
 WRITELN('            *                                               *');
 WRITELN('            ************ SELECT EMPHASIS MENU ***************');
 WRITELN;
 READLN(INPT,EMPCODE);
 WRITELN(EMPCODE);           (* ECHO PRINT *)
 WRITELN;
      IF EMPCODE IN (.'C','c','I','i','T','t','D','d'.) THEN
         ok := true
      else
         begin    (* else *)
            WRITELN('SORRY, WRONG INPUT, TRY AGAIN');
            writeln;
            PAGE;  (* CLEAR SCREEN *)
         end;     (* else *)
   UNTIL OK;  (* END REPEAT LOOP *)

   PAGE;  (* CLEAR SCREEN *)

   if empcode in (.'C','c'.) then
      begin  (* if c *)
         WRITELN;
         WRITELN;
         WRITELN;
         WRITE('YOU HAVE SELECTED "COMPUTER CENTER AND NETWORK ');
         WRITELN('OPERATIONS" AS YOUR EMPHASIS AREA');
         writeln;
      END    (* IF C *)
   else
```

```
    IF EMPCODE IN (.'I','i'.) THEN
       begin  (* if i *)
          WRITELN;
          WRITELN;
          WRITELN;
          WRITE('    ');
          WRITE('YOU HAVE SELECTED "INFORMATION AND COMPUTER ');
          WRITELN('NETWORKS" AS YOUR EMPHASIS AREA');
          writeln;
       END    (* IF I *)
    else
       IF EMPCODE IN (.'T','t'.) THEN
          begin  (* if t *)
             WRITELN;
             WRITELN;
             WRITELN;
             WRITE('          ');
             WRITE('YOU HAVE SELECTED "TACTICAL SYSTEMS" ');
             WRITELN('AS YOUR EMPHASIS AREA');
             writeln;
          END    (* IF T *)
       else
          IF EMPCODE IN (.'D','d'.) THEN
             begin  (* if d *)
                WRITELN;
                WRITELN;
                WRITELN;
                WRITE('     ');
                WRITE('YOU HAVE SELECTED "DECISION SUPPORT ');
                WRITELN('SYSTEMS" AS YOUR EMPHASIS AREA');
                writeln;
             end;   (* if D *)
WRITELN;
WRITE('               ');
WRITELN('DO YOU WANT TO LIST EMPHASIS AREA COURSES NOW? (Y/N)');
READLN(INPT,LOOK);
WRITELN(LOOK);          (* ECHO *)
IF LOOK IN (.'Y','y'.) THEN
   BEGIN
      PAGE;
      LISTEMP;   (* CALL PROCEDURE *)
   END
ELSE
   PAGE;
WRITELN;
WRITELN;
WRITELN;
WRITELN;
```

```
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN;
    WRITE('                    ');
    WRITELN('DO YOU WANT TO ADD EMPHASIS AREA COURSES NOW? (Y/N)');
    READLN(INPT,NOW);
    WRITELN(NOW);          (* ECHO *)
    IF NOW IN ['Y','y'] THEN
       BEGIN
          PAGE;
          ADD_A_COURSE;      (* CALL PROCEDURE *)
       END
    ELSE
       PAGE;
END;     (* PROCEDURE *)

{*************************************************************************}
{* CHECK IF EMPHASIS AREA HAS BEEN SELECTED AND COURSES SCHEDULED        *}
{*************************************************************************}

PROCEDURE EMP_CHECK;

VAR
   N           : INTEGER;  (* INDEX *)
   EMPCOUNT    : INTEGER;

BEGIN  (* PROCEDURE *)

  EMPCOUNT := 0;      (* INITIALIZE *)
  IF EMPCODE IN ['C','c','I','i','T','t','D','d'] THEN
    BEGIN  (* IF EMPCODE *)
      IF EMPCODE IN ['C','c'] THEN
        BEGIN  (* IF C *)
        FOR N := 1 TO NR_COURSES DO
          BEGIN  (* FOR N *)
          WITH DATA_FILE[N] DO
            BEGIN  (* WITH *)
            IF CCNO = 2 THEN
              BEGIN  (* IF 2 *)
                IF TERM = 0 THEN
                  BEGIN  (* IF 0 *)
                    WRITE(COURSEID,COURSENR,' MUST BE SCHEDULED FOR ');
                    WRITELN('"COMPUTER CENTER AND NETWORK OPERATIONS"');
                  END    (* IF 0 *)
                ELSE
                  BEGIN  (* ELSE *)
                    WRITE('YOUR EMPHASIS AREA REQUIRED COURSE');
```

```
                WRITELN(' HAS BEEN SCHEDULED OR VALIDATED');
              END;    (* ELSE *)
          END     (* IF 2 *)
        ELSE
        IF CCNO = 1 THEN
          BEGIN  (* IF 1 *)
            IF TERM > 0 THEN
              EMPCOUNT := EMPCOUNT + 1;
          END;    (* IF 1 *)
        END;    (* WITH *)
      END;    (* FOR N *)
    END     (* IF C *)
  ELSE
  IF EMPCODE IN ['I','i'] THEN
    BEGIN  (* IF I *)
    FOR N := 1 TO NR_COURSES DO
      BEGIN  (* FOR N *)
      WITH DATA_FILE[N] DO
        BEGIN  (* WITH *)
        IF ICN  = 2 THEN
          BEGIN  (* IF 2 *)
            IF TERM = 0 THEN
              BEGIN  (* IF 0 *)
                WRITE(COURSEID,COURSENR,' MUST BE SCHEDULED FOR ');
                WRITELN('"INFORMATION AND COMPUTER NETWORKS"');
              END     (* IF 0 *)
            ELSE
              BEGIN  (* ELSE *)
                WRITE('YOUR EMPHASIS AREA REQUIRED COURSE');
                WRITELN(' HAS BEEN SCHEDULED OR VALIDATED');
              END;    (* ELSE *)
          END     (* IF 2 *)
        ELSE
        IF ICN  = 1 THEN
          BEGIN  (* IF 1 *)
            IF TERM > 0 THEN
              EMPCOUNT := EMPCOUNT + 1;
          END;    (* IF 1 *)
        END;    (* WITH *)
      END;    (* FOR N *)
    END     (* IF I *)
  ELSE
  IF EMPCODE IN ['T','t'] THEN
    BEGIN  (* IF T *)
    FOR N := 1 TO NR_COURSES DO
      BEGIN  (* FOR N *)
      WITH DATA_FILE[N] DO
        BEGIN  (* WITH *)
```

```
            IF TS    = 2 THEN
              BEGIN  (* IF 2 *)
                IF TERM = 0 THEN
                  BEGIN  (* IF 0 *)
                    WRITE(COURSEID,COURSENR,' MUST BE SCHEDULED FOR ');
                    WRITELN('"TACTICAL SYSTEMS"');
                  END    (* IF 0 *)
                ELSE
                  BEGIN  (* ELSE *)
                    WRITE('YOUR EMPHASIS AREA REQUIRED COURSE');
                    WRITELN(' HAS BEEN SCHEDULED OR VALIDATED');
                  END;   (* ELSE *)
              END    (* IF 2 *)
            ELSE
            IF TS    = 1 THEN
              BEGIN  (* IF 1 *)
                IF TERM > 0 THEN
                  EMPCOUNT := EMPCOUNT + 1;
              END;   (* IF 1 *)
            END;   (* WITH *)
          END;   (* FOR N *)
        END    (* IF T *)
      ELSE
      IF EMPCODE IN ['D','d'] THEN
        BEGIN  (* IF D *)
        FOR N := 1 TO NR_COURSES DO
          BEGIN  (* FOR N *)
          WITH DATA_FILE[N] DO
            BEGIN  (* WITH *)
            IF DSS   = 2 THEN
              BEGIN  (* IF 2 *)
                IF TERM = 0 THEN
                  BEGIN  (* IF 0 *)
                    WRITE(COURSEID,COURSENR,' MUST BE SCHEDULED FOR ');
                    WRITELN('"DECISION SUPPORT SYSTEMS"');
                  END    (* IF 0 *)
                ELSE
                  BEGIN  (* ELSE *)
                    WRITE('YOUR EMPHASIS AREA REQUIRED COURSE');
                    WRITELN(' HAS BEEN SCHEDULED OR VALIDATED');
                  END;   (* ELSE *)
              END    (* IF 2 *)
            ELSE
            IF DSS   = 1 THEN
              BEGIN  (* IF 1 *)
                IF TERM > 0 THEN
                  EMPCOUNT := EMPCOUNT + 1;
              END;   (* IF 1 *)
```

```
            END;    (* WITH *)
          END;    (* FOR N *)
        END;    (* IF D *)

        IF EMPCOUNT < MINEMP THEN
          BEGIN  (* COUNT < MIN *)
            WRITELN;
            WRITE('YOU HAVE SCHEDULED ',EMPCOUNT:2,' EMPHASIS');
            WRITELN(' AREA ELECTIVE(S).  A MINIMUM OF 3 IS REQUIRED');
            WRITELN;
          END    (* COUNT < MIN *)
        ELSE
          BEGIN  (* ELSE *)
            WRITELN;
            WRITELN('YOU HAVE SCHEDULED ENOUGH EMPHASIS AREA ELECTIVES');
            WRITELN;
          END;    (* ELSE *)
    END    (* IF EMPCODE *)
  ELSE
    BEGIN  (* ELSE *)
      WRITELN;
      WRITELN('AN EMPHASIS AREA HAS NOT BEEN SELECTED YET');
    END;    (* ELSE *)

END;    (* PROCEDURE *)

{*************************************************************************}
{* ASSIGN A 1 TO "TERM" FIELD OF VALIDATED COURSES                       *}
{*************************************************************************}

PROCEDURE VALIDATE;

CONST
    FACTOR     = 64;

VAR
    ANSWER      : CHAR;
    ID          : COURSECODE;
    NR          : COURSENUMB;
    S,K,T       : INTEGER;
    DONE,FOUND  : BOOLEAN;
    NONE        : BOOLEAN;

BEGIN  (* PROCEDURE *)

    DONE := FALSE;     (* INITIALIZE *)
    REPEAT    (* UNTIL DONE *)
      FOUND := FALSE;       (* INITIALIZE *)
```

```
WRITELN;
WRITELN;
WRITELN;
WRITELN;
WRITE('                ENTER THE SIX CHARACTER CODE OF THE COURSE TO ');
WRITELN('VALIDATE');
READLN(INPT,ID,NR);
BEGIN  (* CONVERT *)
 FOR K := 1 TO STRLEN(ID) DO
  IF ID(.K.) IN (.'a'..'z'.) THEN
    ID(.K.) := CHR(ORD(ID(.K.)) + FACTOR);
END;   (* CONVERT *)
WRITELN(ID:8,NR);    (* ECHO PRINT COURSE *)
FOR S := 1 TO NR_COURSES DO
          BEGIN  (* FOR S *)
             WITH DATA_FILE(.S.) DO
                BEGIN  (* WITH *)
                   IF ID = COURSEID THEN
                      IF NR = COURSENR THEN
                         BEGIN                (* IF NR *)
                         TERM := 1;           (* INDICATES VALIDATION *)
                         FOUND := TRUE;
                         END;                 (* IF NR *)
                END;   (* WITH *)
          END;   (* FOR S *)
       IF NOT FOUND THEN
          BEGIN  (* IF NOT *)
             WRITELN;
             WRITE('             ');
             WRITE(ID,NR,' IS NOT LOCATED IN THE DATABASE.');
             WRITELN('CHECK FOR INPUT ERROR');
          END;   (* IF NOT *)

WRITELN;
WRITE('                    ');
WRITELN('DO YOU WISH TO VALIDATE ANY OTHER COURSES?');
WRITE('                    ');
WRITELN('ENTER Y FOR YES / ANY OTHER CHARACTER FOR NO');
WRITELN;
READLN(INPT,ANSWER);
WRITELN(ANSWER);              (* ECHO PRINT THE INPUT *)

IF ANSWER IN (.'Y','y'.) THEN
 BEGIN  (* IF Y *)
   DONE := FALSE;
   PAGE;  (* CLEAR SCREEN *)
 END    (* IF Y *)
ELSE
```

```
        BEGIN (* ELSE *)
          DONE := TRUE;
          PAGE;    (* CLEAR SCREEN *)
        END;  (* ELSE *)
    UNTIL DONE;       (* END OF REPEAT LOOP *)
    PAGE;  (* CLEAR SCREEN *)
    WRITELN;
    WRITELN('VALIDATED COURSES ********************************');
    WRITELN;
    NONE := TRUE;        (* INITIALIZE *)
    FOR T := 1 TO NR_COURSES DO
       BEGIN  (* FOR T *)
       WITH DATA_FILE(.T.) DO
          BEGIN  (* WITH *)
          IF TERM = 1 THEN
             BEGIN  (* IF *)
                WRITELN(COURSEID:8,COURSENR);
                WRITELN;
                NONE := FALSE;
             END;    (* IF *)
          END;    (* WITH *)
       END;    (* FOR T *)
    IF NONE = TRUE THEN
      BEGIN
       WRITELN('       NONE');
       WRITELN;
      END;
    WRITELN; (**)
    WRITELN;  (**)
    WRITELN;    (**)
    WRITELN;     (**)
    WRITELN;      (**)
    WRITELN;       (**)
    WRITELN; {*************************************************************}
    WRITELN; {* WRITELN'S USED TO PUSH MAIN MENU TO NEXT FULL SCREEN    *}
    WRITELN; {*************************************************************}
    WRITELN;       (**)
    WRITELN;      (**)
    WRITELN;     (**)
    WRITELN;    (**)
    WRITELN;   (**)
    WRITELN; (**)

END;    (* PROCEDURE *)

{*************************************************************************}
{*  SEE IF MINIMUM HOUR REQUIREMENTS HAVE BEEN MET BY THE SCHEDULE     *}
{*************************************************************************}
```

```
PROCEDURE MIN_HRS_CHECK;

VAR
   I            : INTEGER;   (* INDEX *)
   QTR          : INTEGER;

BEGIN

      WRITELN;
      WRITELN('STATUS OF MINIMUM REQUIREMENT FULFILLMENT');
      WRITELN('*****************************************');
      WRITELN;
      BEGIN  (* AS IF *)
      IF ASHRS < MINASHRS THEN
         BEGIN
           ASDEF := MINASHRS - ASHRS;
           WRITE('YOU NEED TO SCHEDULE AT LEAST',ASDEF:3);
           WRITELN(' MORE HOURS OF ADMIN SCIENCE CLASSES');
           WRITELN;
         END;
      END;   (* AS IF *)


      BEGIN  (* CS IF *)
      IF CSHRS < MINCSHRS THEN
         BEGIN
            CSDEF := MINCSHRS - CSHRS;
            WRITE('YOU NEED TO SCHEDULE AT LEAST',CSDEF:3);
            WRITELN(' MORE HOURS OF COMPUTER SCIENCE CLASSES');
            WRITELN;
         END;
      END;   (* CS IF *)


      BEGIN  (* GRAD IF *)
      IF GRADHRS < MINGRADHRS THEN
         BEGIN
            GRADDEF := MINGRADHRS - GRADHRS;
            WRITE('YOU NEED TO SCHEDULE AT LEAST',GRADDEF:3);
            WRITELN(' MORE GRADUATE LEVEL COURSES (3000 OR 4000 LEVEL');
            WRITELN;
         END;
      END;   (* GRAD IF *)


      BEGIN  (* LVL4 IF *)
      IF LVL4HRS < MINLVL4HRS THEN
```

```
            BEGIN
               LVL4DEF := MINLVL4HRS - LVL4HRS;
               WRITE('YOU NEED TO SCHEDULE AT LEAST',LVL4DEF:3);
               WRITELN(' MORE HOURS OF 4000 LEVEL COURSES');
               WRITELN;
            END;
         END;   (* LVL4 IF *)


         QTR := 0;          (* INITIALIZE *)
         FOR I := 2 TO 7 DO
            BEGIN (* FOR I *)
            QTR := I - 1;
            IF NR_CLASSES(.I.) = TRUE THEN
               BEGIN
                  WRITE('YOU HAVE NOT SCHEDULED ENOUGH CLASSES ');
                  WRITELN('FOR QTR',QTR:2);
                  IF QTR < 4 THEN
                    BEGIN  (* QTR < 4 *)
                      WRITE('YOU WILL NEED APPROVAL TO TAKE LESS THAN');
                      WRITELN(' THE MINIMUM OF 4 COURSES THIS TERM');
                      WRITELN;
                    END    (* QTR < 4 *)
                  ELSE
                  IF QTR > 3 THEN
                    BEGIN  (* QTR > 3 *)
                      WRITE('YOU WILL NEED APPROVAL TO TAKE LESS THAN');
                      WRITELN(' THE MINIMUM OF 3 COURSES THIS TERM');
                      WRITELN;
                    END;   (* QTR > 3 *)
               END;
            END;  (* FOR I *)

      END;  (* PROCEDURE *)

      {*****************************************************************}
      {* UPDATE MIN REQUIREMENT COUNTERS AND SHOW SELECTED SCHEDULE     *}
      {*****************************************************************}

      PROCEDURE UPDATE_SHOW;

      VAR
         BAD,NONE    : BOOLEAN;
         X,Y,A,T     :INTEGER;  (* INDEX VARIABLE *)
         Z           :INTEGER;  (* INDEX VARIABLE/QUARTER NUMBER *)
         COUNT       :INTEGER;  (* COUNT NUMBER OF CLASSES PER TERM *)

      BEGIN    (* PROCEDURE *)
```

```
ASHRS     := 0;                                    {**}
CSHRS     := 0;                                    {**}
LVL4HRS   := 0;{* LEVEL 4000 *)                   {**}
GRADHRS   := 0;{* LEVEL 3000 + 4000 *)            {**************}
TOTALHRS  := 0;                                    {* INITIALIZE *}
ASDEF     := 0;                                    {* VARIABLES  *}
CSDEF     := 0;                                    {**************}
GRADDEF   := 0;                                    {**}
LVL4DEF   := 0;                                    {**}
Z         := 0;                                    {**}
FOR X := 2 TO 7 DO
   BEGIN  (* FOR X *)
   NR_CLASSES(.X.) := FALSE;
   COUNT := 0;
   Z := X - 1;
   WRITELN;
   WRITE('  *********');
   WRITE('**************** CLASSES FOR QTR ',Z:1,' *************');
   WRITELN('***************');
   WRITELN;
   FOR Y := 1 TO NR_COURSES DO
      BEGIN  (* FOR Y *)
      WITH DATA_FILE(.Y.) DO
         BEGIN (* WITH *)
         IF TERM = X THEN
            BEGIN  (* IF TERM = X *)
            COUNT := COUNT + 1;
            WRITE('  ');
            WRITELN(DESCRIPT(.1.));
            TOTALHRS := TOTALHRS + HOURS;
               BEGIN  (* LOOP 1 *)
               IF CURRIC = 'AS' THEN
                  ASHRS := ASHRS + HOURS
               ELSE
                  IF CURRIC = 'CS' THEN
                     CSHRS := CSHRS + HOURS
               END;  (* LOOP 1 *)
               BEGIN (* LOOP 2 *)
                  IF COURSENR(.1.) = '4' THEN
                     BEGIN
                     LVL4HRS := LVL4HRS + HOURS;
                     GRADHRS := GRADHRS + HOURS;
                     END
                  ELSE
                     IF COURSENR(.1.) = '3' THEN
                        GRADHRS := GRADHRS + HOURS;
               END;  (* LOOP 2 *)
```

```
                END; (* IF TERM = X *)
            END; (* WITH *)
        END; (* FOR Y *)

        IF X < 5 THEN     (* TERM 1 - 3 *)
            IF COUNT < 4 THEN
                NR_CLASSES(.X.) := TRUE;
        IF X > 4 THEN     (* TERM 4 - 6 *)
            IF COUNT < 3 THEN
                NR_CLASSES(.X.) := TRUE;

    END; (* FOR X *)

    WRITELN;
    WRITELN('CUMULATIVE STATISTICS AFTER QTR ',Z:1);
    WRITELN;
    WRITELN('    ADMIN SCIENCE HOURS      = ',ASHRS:3);
    WRITELN('    COMPUTER SCIENCE HOURS   = ',CSHRS:3);
    WRITELN('    GRADUATE LEVEL HOURS     = ',GRADHRS:3);
    WRITELN('    4000 LEVEL HOURS         = ',LVL4HRS:3);
    WRITELN('    TOTAL HOURS              = ',TOTALHRS:3);

MIN_HRS_CHECK;    (* CALL PROCEDURE *)

WRITELN;
WRITELN('VALIDATED COURSES ********************************');
WRITELN;
NONE := TRUE;          (* INITIALIZE *)
FOR T := 1 TO NR_COURSES DO
    BEGIN  (* FOR T *)
    WITH DATA_FILE(.T.) DO
        BEGIN  (* WITH *)
        IF TERM = 1 THEN
            BEGIN  (* IF *)
                WRITELN(COURSEID:8,COURSENR);
                WRITELN;
                NONE := FALSE;
            END;   (* IF *)
        END;   (* WITH *)
    END;   (* FOR T *)
IF NONE = TRUE THEN
  BEGIN
    WRITELN('       NONE');
    WRITELN;
  END;

BEGIN  (* SHOW REQD COURSES *)
 WRITE('THE FOLLOWING REQUIRED COURSES HAVE NOT BEEN SCHEDULED');
```

```
    WRITELN(' OR VALIDATED: ');
    WRITELN;
    BAD := FALSE;     (* INITIALIZE *)
    FOR A := 1 TO NR_COURSES DO
      BEGIN  (* FOR A *)
       WITH DATA_FILE(.A.) DO
          BEGIN  (* WITH *)
             IF REQD = 1 THEN
                IF TERM < 1 THEN
                  BEGIN  (* IF TERM *)
                   BAD := TRUE;
                   WRITELN(COURSEID:8,COURSENR:4);
                  END;   (* IF TERM *)
          END;    (* WITH *)
      END;    (* FOR A *)
    IF BAD = FALSE THEN
      BEGIN  (* IF NOT BAD *)
       WRITELN('       NONE');
       WRITELN;
      END;    (* IF NOT BAD *)
   END;   (* SHOW REQD COURSES *)

    EMP_CHECK;          (* CALL PROCEDURE *)

    PREREQUISITE_CHECK;  (* CALL PROCEDURE *)

END;  (* PROCEDURE *)

{*************************************************************************}
{* DROP A COURSE FROM THE SCHEDULE                                       *}
{*************************************************************************}

PROCEDURE DROP_A_COURSE;

CONST
    FACTOR     = 64;

VAR
    FOUND,HOLD : BOOLEAN;
    DRPANS     : CHAR;
    ID         : COURSECODE;
    NR         : COURSENUMB;
    C,K        : INTEGER;       (* INDEX *)
    STILL      : CHAR;

BEGIN  (* PROCEDURE *)

  HOLD := TRUE;
```

```
REPEAT  (* UNTIL NOT HOLD *)
 FOUND := FALSE;     (* INITIALIZE *)
 WRITELN;
 WRITELN;
 WRITELN;
 WRITELN;
 WRITELN('        ENTER THE SIX CHARACTER CODE OF THE COURSE TO DROP');
 READLN(INPT,ID,NR);
 BEGIN  (* CONVERT *)
  FOR K := 1 TO STRLEN(ID) DO
   IF ID(.K.) IN (.'a'..'z'.) THEN
     ID(.K.) := CHR(ORD(ID(.K.)) + FACTOR);
 END;    (* CONVERT *)
 WRITELN(ID:8,NR);       (* ECHO *)
 FOR C := 1 TO NR_COURSES DO
   BEGIN  (* FOR C *)
     WITH DATA_FILE(.C.) DO
       BEGIN  (* WITH *)
       IF ID = COURSEID THEN
         BEGIN  (* IF ID *)
         IF NR = COURSENR THEN
           BEGIN  (* IF NR *)
           FOUND := TRUE;
           IF TERM = 0 THEN
            BEGIN
             WRITE('                    ');
             WRITELN(ID,NR,' IS NOT ON THE CURRENT SCHEDULE');
            END
           ELSE      (* MEANS IT IS ON THE SCHEDULE *)
             BEGIN  (* ELSE *)
             IF REQD = 1 THEN
               BEGIN  (* IF REQD *)
               WRITE('                    ');
               WRITELN(ID,NR,' IS A REQUIRED COURSE');
               WRITE('                    ');
               WRITELN('DO YOU STILL WISH TO DROP IT?');
               WRITE('                    ');
               WRITELN('ENTER "Y" FOR YES, ANY OTHER FOR NO');
               READLN(INPT,STILL);
               WRITELN(STILL);          (* ECHO *)
               WRITELN;
               PAGE;   (* CLEAR SCREEN *)
               IF STILL IN (.'Y','y'.) THEN
                 BEGIN  (* IF Y *)
                 TERM := 0;
                 WRITELN;
                 WRITELN;
                 WRITELN;
```

```
                    WRITELN;
                    WRITE('                    ');
                    WRITELN(ID,NR,' HAS BEEN DROPPED FROM THE SCHEDULE');
                    END    (* IF Y *)
                  ELSE
                   BEGIN
                    WRITELN;
                    WRITELN;
                    WRITELN;
                    WRITE('                              ');
                    WRITELN(ID,NR,' WILL REMAIN ON THE SCHEDULE');
                   END;
                  END    (* IF REQD *)
                ELSE  (* REQD = 0 / NOT REQD *)
                  BEGIN  (* ELSE/REQD = 0 *)
                    TERM := 0;
                    WRITELN;
                    WRITELN;
                    WRITELN;
                    WRITELN;
                    WRITE('                    ');
                    WRITELN(ID,NR,' HAS BEEN DROPPED FROM THE SCHEDULE');
                  END;    (* ELSE/REQD = 0 *)
                END;    (* ELSE-MEANS ON SCHEDULE *)
              END;    (* IF NR *)
            END;    (* IF ID *)
          END;    (* WITH *)
        END;    (* FOR C *)
      IF NOT FOUND THEN
       BEGIN
        WRITE('                    ');
        WRITELN(ID,NR,' IS NOT LOCATED IN THE DATABASE');
       END;

       WRITELN;
       WRITELN;
       WRITELN;
       WRITELN;
       WRITE('                    ');
       WRITELN('DO YOU WANT TO DROP ANOTHER COURSE?');
       WRITE('                    ');
       WRITELN('ENTER "Y" FOR YES. ANY OTHER CHARACTER FOR NO');
       WRITELN;
       READLN(INPT,DRPANS);
       WRITELN(DRPANS);      (* ECHO PRINT *)
       IF DRPANS IN ['Y','y'] THEN
        BEGIN (* IF Y *)
         HOLD := TRUE;
```

```
      PAGE;  (* CLEAR SCREEN *)
    END   (* IF Y *)
  ELSE
    HOLD := FALSE;

UNTIL NOT HOLD;   (* END REPEAT LOOP *)
PAGE;  (* CLEAR SCREEN *)

END;   (* PROCEDURE *)

{*************************************************************************}
{* PROVIDE A HELP MENU FOR THE USER                                      *}
{*************************************************************************}

PROCEDURE MAIN_MENU_DISPLAY;

BEGIN  (* PROCEDURE *)

WRITELN;
WRITELN;
WRITE('     ');
WRITELN('      ********************* MAIN MENU ************************');
WRITE('     ');
WRITELN('      *                                                      *');
WRITE('     ');
WRITELN('      *    ENTER "H" FOR HELP AND GENERAL INFORMATION        *');
WRITE('     ');
WRITELN('      *    ENTER "V" TO VALIDATE A COURSE                    *');
WRITE('     ');
WRITELN('      *    ENTER "A" TO ADD  A COURSE TO   THE SCHEDULE      *');
WRITE('     ');
WRITELN('      *    ENTER "D" TO DROP A COURSE FROM THE SCHEDULE      *');
WRITE('     ');
WRITELN('      *    ENTER "E" TO SELECT AN EMPHASIS AREA              *');
WRITE('     ');
WRITELN('      *    ENTER "L" TO LIST COURSES AVAILABLE               *');
WRITE('     ');
WRITELN('      *    ENTER "U" TO SEE YOUR UPDATED SCHEDULE AND        *');
WRITE('     ');
WRITELN('      *              TO SEE YOUR STATUS ON FULFILLMENT       *');
WRITE('     ');
WRITELN('      *              OF MINIMUM REQUIREMENTS                 *');
WRITE('     ');
WRITELN('      *    ENTER "P" TO SEE IF PREREQUISITES HAVE BEEN MET   *');
WRITE('     ');
WRITELN('      *    ENTER "Q" TO QUIT                                 *');
WRITE('     ');
WRITELN('      *                                                      *');
```

```
WRITE('     ');
WRITELN('     ********************* MAIN MENU ************************');

END;   (* PROCEDURE *)


{*************************************************************************}
{* PROVIDE GENERAL GUIDANCE TO THE USER                                  *}
{*************************************************************************}

PROCEDURE GUIDANCE;

BEGIN  (* PROCEDURE *)

WRITELN('                                                                ');
WRITELN('                                                                ');
WRITELN('   Debug? - If this should appear in the top left  corner       ');
WRITELN('            of your screen , you have made an input error       ');
WRITELN('            that has terminated the program.                    ');
WRITELN('            Enter "q" (without quotes) to  exit the debug       ');
WRITELN('            routine and re-enter CMS.                           ');
WRITELN('                                                                ');
WRITELN('   MORE...- If  this  should  appear in  the lower  right       ');
WRITELN('            corner of your screen,  you must depress the        ');
WRITELN('            "ALT" key  and the "PA2" key  (or the "CLEAR"       ');
WRITELN('            key) simultaneously to see the next screen of       ');
WRITELN('            information.                                        ');
WRITELN('            NEVER  try to enter data  when this signal is       ');
WRITELN('            on your screen: it WILL terminate the program.      ');
WRITELN('                                                                ');
WRITELN('                                                                ');
WRITELN('                                                                ');
WRITELN('                                                                ');
WRITELN('                                                                ');
WRITELN('                                                                ');
WRITELN('                                                                ');
WRITELN('                                                                ');
WRITELN('                                                                ');
WRITELN('      THIS PROGRAM IS DESIGNED TO HELP COMPUTER SYSTEM MGMT     ');
WRITELN('    STUDENTS (CURRIC #367) PREPARE THEIR ACADEMIC SCHEDULES.    ');
WRITELN('    EACH STUDENT IS PROVIDED  WITH  A  PRESET  SCHEDULE THAT    ');
WRITELN('    CONTAINS ALL REQUIRED COURSES.    THE STUDENT MUST BUILD    ');
WRITELN('    ON THAT  SCHEDULE IN  ORDER TO MEET  MINIMUM  GRADUATION    ');
WRITELN('    REQUIREMENTS:                                               ');
WRITELN('          CATEGORY                          MINIMUM             ');
WRITELN('          ADMIN SCIENCE HOURS                  24               ');
WRITELN('          COMPUTER SCIENCE HOURS               16               ');
WRITELN('          GRADUATE LEVEL HOURS                 48               ');
```

```
WRITELN('          4000 LEVEL HOURS                    12 (OF THE 48 GRAD)');
WRITELN('          COURSES PER TERM (1-3)               4');
WRITELN('                "     "    (4-6)               3');
WRITELN('   PLUS:');
WRITELN('      1. SCHEDULE OR VALIDATE ALL REQUIRED COURSES');
WRITELN('      2. SELECT AN EMPHASIS AREA WITH:');
WRITELN('            A. ONE REQUIRED COURSE');
WRITELN('            B. THREE ELECTIVES');
WRITELN(' ');
WRITELN(' ');
WRITELN(' ');
WRITELN('   THIS MUST ALL BE DONE KEEPING IN MIND THAT ALL COURSES');
WRITELN('   ARE NOT OFFERED EVERY QUARTER.     FURTHERMORE, COURSE');
WRITELN('   PREREQUISITES MUST BE SCHEDULED OR VALIDATED.');
WRITELN(' ');
WRITELN('   THIS PROGRAM WILL KEEP TRACK OF ALL THESE REQUIREMENTS');
WRITELN('   FOR YOU AND  WILL ALLOW YOU TO CREATE ANY SCHEDULE YOU');
WRITELN('   DESIRE (WITHIN DATABASE LIMITATIONS).');
WRITELN(' ');
WRITELN('   TO OBTAIN A HARD COPY OF SCHEDULE CREATED:');
WRITELN('      A COPY OF THE LATEST ENTIRE  SESSION ON  "SKEDULER"');
WRITELN('      WILL BE  RESIDENT ON YOUR  A-DISK  UNDER  THE  FILE');
WRITELN('      "SKEDULER LISTING".     THAT  FILE CAN BE EDITED IN');
WRITELN('      XEDIT AND PRINTED ON THE VM PRINTER.   IF YOUR LAST');
WRITELN('      SESSION WAS OF CONSIDERABLE LENGTH, YOU MAY HAVE TO');
WRITELN('      "DEFINE STORAGE 1M"  PRIOR TO  GOING INTO  XEDIT OR');
WRITELN('      YOU  WILL  BE TOLD  THAT YOUR  LISTING  FILE IS TOO');
WRITELN('      LARGE.');
WRITELN(' ');
WRITELN(' ');
WRITELN(' ');
WRITELN(' ');

END;   (* PROCEDURE *)

$PAGE;
{**************************************************************}
{*                    BEGIN MAIN PROGRAM                      *}
{**************************************************************}

BEGIN (* MAIN PROGRAM *)
PAGE; (* SEPERATE PROGRAM FROM OUTPUT *)

  EMPCODE := ' ';          (* INITIALIZE *)

  GET_DATA(DATA_FILE);     (* CALL PROCEDURE *)
```

```
RESET (INPT,'TERMINAL'); (* TO   COME INTERACTIVE W/ TERMINAL *)

MATCH_TERM_TO_QTR;      (* CALL PROCEDURE *)

THRU := FALSE;          (* INITIALIZE *)
REPEAT  (* UNTIL THRU *)
  GOOD := FALSE;        (* INITIALIZE *)
  REPEAT  (* UNTIL GOOD *)
    WRITELN;
    MAIN_MENU_DISPLAY;  (* CALL PROCEDURE *)
    READLN(INPT,MENU);
    WRITELN(MENU);      (* ECHO *)
    IF MENU IN (.'Q','q','A','a','D','d','V','v','E','e'.) THEN
      GOOD := TRUE
    ELSE
    IF MENU IN (.'L','l','U','u','P','p','H','h'.) THEN
      GOOD := TRUE
     ELSE
       PAGE;

  UNTIL GOOD;  (* END OF REPEAT LOOP *)

  PAGE;  (* CLEAR SCREEN *)

  IF MENU IN (.'Q','q'.) THEN
    THRU := TRUE
  ELSE
  IF MENU IN (.'A','a'.) THEN
    ADD_A_COURSE
  ELSE
  IF MENU IN (.'D','d'.) THEN
    DROP_A_COURSE
  ELSE
  IF MENU IN (.'V','v'.) THEN
    BEGIN  (* IF V *)
      VALIDATE;
    END    (* IF V *)
  ELSE
  IF MENU IN (.'E','e'.) THEN
    EMPHASIS_AREA
  ELSE
  IF MENU IN (.'L','l'.) THEN
    LISTER
  ELSE
  IF MENU IN (.'U','u'.) THEN
    UPDATE_SHOW
  ELSE
  IF MENU IN (.'P','p'.) THEN
```

```
      PREREQUISITE_CHECK
    ELSE
    IF MENU IN (.'H','h'.) THEN
      GUIDANCE

  UNTIL THRU;  (* END REPEAT LOOP *)

PAGE;(* SEPERATE OUTPUT FROM EXECUTION STATEMENT *)
end. (* of main program *)
(*************************************************************)
(*                    END MAIN PROGRAM                       *)
(*************************************************************)
```

APPENDIX I

INTEGRATION TEST SPECIFICATION

1. SCOPE

   1.1 "SKEDULER" is an interactive, terminal oriented, menu driven program. Because of these programming objectives, effective use of menus, efficient use of screen space for data display, ease of user interactions, and data examination capabilities are critical functional characteristics that must be present and thoroughly tested.

   1.2 Modules will be integrated using top down and incremental techniques. The top down approach (starting with superordinate modules and working down using stubs) will be used instead of the bottom up approach (starting with the most subordinate modules and working up using drivers) because it provides a preliminary view of the entire program from the very beginning. The incremental approach approach (new modules added one at a time) will be used instead of the phased approach (all modules are combined simultaneously) because it has the advantage of isolating the source of new errors by focusing on the newly added module.

2. TEST PLAN

   2.1 INCREMENTS

      2.1.1 Program Control and Data Base Structure

      2.1.2 Transaction Dispatchers

      2.1.3 Display Format and Generation

      2.1.4 User Interaction

      2.1.5 Data Examination and Display

2.1.6 Module Addition After Program Completion

2.2 OVERHEAD SOFTWARE

2.2.1 Stubs will be required during each increment to fill in the functions required from uncompleted subordinate modules.

2.3 ENVIRONMENT AND RESOURCES

2.3.1 All mechanized testing will be conducted using the IBM 3033 mainframe computer from IBM 3278 terminals.

3. TEST PROCEDURE

3.1 INCREMENT 1 - PROGRAM CONTROL & DATA BASE STRUCTURE

3.1.1 Order of Integration

3.1.1.1 Skeduler Control Module

3.1.1.2 Get Data

3.1.1.3 Match Term To Qtr

3.1.2 Test Case Design - see appendix 5.1

3.1.3 Stubs Required

3.1.3.1 for Skeduler Control Module

3.1.3.1.1 Get Data

3.1.3.1.2 Match Term To Qtr

3.1.3.1.3 Main Menu Switch

3.2 INCREMENT 2 - TRANSACTION DISPATCHERS

3.2.1 Order of Integration

3.2.1.1 Main Menu Switch (w/in Skeduler Control)

3.2.1.2 MAIN Menu Display

3.2.1.3 Lister (List Menu Switch)

3.2.2 Test Case Design - see appendix 5.1

3.2.3 Stubs Required

3.2.3.1 for Main Menu Switch

3.2.3.1.1 Add_A_Course

3.2.3.1.2 Drop_A_Course

3.2.3.1.3 Validate

3.2.3.1.4 Emphasis_Area

3.2.3.1.5 Lister
3.2.3.1.6 Update_Show
3.2.3.1.7 Prerequisite_Check
3.2.3.1.8 Guidance
3.2.3.2 for Lister
3.2.3.2.1 LISTAS
3.2.3.2.2 LISTCS
3.2.3.2.3 LIST3
3.2.3.2.4 LIST4
3.2.3.2.5 LISTEMP
3.2.3.2.6 LIST_Required_Courses
3.2.3.2.7 LISTIND
3.2.3.2.8 LISTALL
3.2.3.2.9 Names

3.3 INCREMENT 3 - DISPLAY FORMAT AND GENERATION
3.3.1 Order of Integration
3.3.1.1 Guidance
3.3.1.2 LIST3
3.3.1.3 LIST4
3.3.1.4 LISTALL
3.3.1.5 LISTAS
3.3.1.6 LISTCS
3.3.1.7 LISTEMP
3.3.1.8 LISTIND
3.3.1.9 List_Required_Courses
3.3.1.10 Names
3.3.2 Test Case Design - see appendix 5.1
3.3.3 Stubs Required - none

3.4 INCREMENT 4 - USER INTERACTION
3.4.1 Order of Integration
3.4.1.1 Emphasis_Area
3.4.1.2 Validate
3.4.1.3 Drop_A_Course
3.4.1.4 Add_A_Course

3.4.1.5 Pick_A_New_Term
3.4.2 Test Case Design - see appendix 5.1
3.4.3 Stubs Required
3.4.3.1 for Emphasis_Area
3.4.3.1.1 Add_A_Course
3.4.3.2 for Emphasis_Area
3.4.3.2.1 Pick_A_New_Term

3.5 INCREMENT 5 - DATA EXAMINATION AND DISPLAY
3.5.1 Order of Integration
3.5.1.1 Update_Show
3.5.1.2 Emp_Check
3.5.1.3 Min_Hours_Check
3.5.1.4 Prerequisite_Check
3.5.2 Test Case Design - see appendix 5.1
3.5.3 Stubs Required
3.5.3.1 for Update_Show
3.5.3.1.1 Emp_Check
3.5.3.1.2 Min_Hours_Check
3.5.3.1.3 Prerequisite_Check

3.6 INCREMENT 6 - MODULE ADDED AFTER PROGRAM COMPLETION
3.6.1 Order of Integration
3.6.1.1 Describe
3.6.2 Test Case Design - see appendix 5.1
3.6.3 Stubs Required - none

4. REFERENCES
4.1 DEUTSCH,M.S., SOFTWARE VERIFICATION AND VALIDATION, PRENTICE-HALL, INC., 1982
4.2 MYERS,G.J., THE ART OF SOFTWARE TESTING, JOHN WILEY & SONS, INC., 1979
4.3 PRESSMAN,R.S., SOFTWARE ENGINEERING, McGRAW-HILL, INC., 1982

5. APPENDIX
5.1 CHART OF MODULE TEST CASES

APPENDIX 5.1

CHART OF MODULE TEST CASES

INDEX:

LOGIC COVERAGE (WHITE BOX):

S = STATEMENT COVERAGE

D/C = DECISION/CONDITION COVERAGE

FUNCTIONAL OPERATION (BLACK BOX):

CEG = CAUSE/EFFECT GRAPHING

BA = BOUNDARY ANALYSIS

EP = EQUIVALENCE PARTITIONING

EG = ERROR GUESSING

	WHITE BOX		BLACK BOX			
INCREMENT/MODULE	S	D/C	CEG	BA	EP	EG
1. SKEDULER CONTROL		*	*	*	*	*
GET_DATA	*			*	*	*
MATCH_TERM_TO_QTR		*		*	*	*
2. MAIN MENU SWITCH		*	*			*
MAIN MENU DISPLAY	*				*	*
LISTER		*	*		*	*
3. GUIDANCE	*				*	*
LIST3	*				*	*
LIST4	*				*	*
LISTALL	*				*	*
LISTAS	*				*	*
LISTCS	*				*	*
LISTEMP		*	*	*	*	*
LISTIND		*	*	*	*	*
LIST_REQ_COURSES	*				*	*
NAMES	*				*	*

(CONTINUED)	WHITE BOX		BLACK BOX			
INCREMENT/MODULE	S	D/C	CEG	BA	EP	EG
4. EMPHASIS_AREA		*	*	*	*	*
VALIDATE		*	*	*	*	*
DROP_A_COURSE		*	*	*	*	*
ADD_A_COURSE		*	*	*	*	*
PICK_A_NEW_TERM		*			*	*
5. UPDATE_SHOW		*	*	*	*	*
EMP_CHECK		*	*	*	*	*
MIN_HOURS_CHECK		*		*	*	*
PREREQUISITE_CHECK		*	*	*	*	*
6. DESCRIBE		*	*	*	*	*

APPENDIX J

MAINTENANCE MANUAL

1. GENERAL

   1.1 Maintenance is divided into four major activities: corrective maintenance, adaptive maintenance, perfective maintenance, and preventitive maintenance. Corrective maintenance deals with finding the cause of errors and correcting them. Adaptive maintenance modifies software to better fit into its changing environment. Perfective maintenance adds new capabilities and enhances present capabilities. Preventative maintenance changes the software to improve future maintainability and reliability.

   1.2 This manual will deal with adaptive maintenance only; however, the design of the program and the existence of its design documentation will allow all of the maintenance activities to be easily implemented when desired.

2. WHEN CHANGES WILL/MAY BE REQUIRED:

   2.1 WHEN A NEW CATALOG IS ISSUED (ANNUALLY)
      2.1.1 new course descriptions
      2.1.2 new hours breakdown (lecture/lab)
      2.1.3 new prerequisites
      2.1.4 new course name or number

   2.2 WHEN REQUIRED COURSE MATRIX IS MODIFIED
      (see Curricular Officer)
      2.2.1 addition/deletion of courses
      2.2.2 movement of courses to different terms
      2.2.3 change in number of course required per term

2.3 WHEN EMPHASIS AREA COURSES ARE ADJUSTED
(see Curricular Officer)

2.3.1 change in course required per emphasis area

2.3.2 change in courses allowed as electives per emphasis area

2.4 WHEN COURSE TENTATIVE SCHEDULE IS MODIFIED (see Tentative Course Schedule for annual changes and see Department Heads for changes that occur during the year)

2.4.1 changes to academic quarter offered

3. WHAT CHANGES WILL BE REQUIRED:

3.1 ADDING (DELETING) A COURSE

3.1.1 All course data is listed in 2 sets (see enclosure 1). Each set is in the same sequence (alphabetic by course ID and then numeric be course number). When a course is added (or deleted) corresponding data must be added (deleted) from both sets (i.e. if a course is added in the 4th row of the set 1 sequence, the description must be added in the 4th row of the set 2 sequence - remembering that set 2 is in blocks of 10 rows per entry, the set 2 entry would be made to rows 31-40).

3.1.2 When adding a course, make sure that you add all prerequisites for the course as well.

3.2 CHANGES TO DATA ITEMS IN SET 1 - These changes can be made simply by putting the program/input file into XEDIT and overlaying the old data with the new data.

4. IMPACT OF DATA BASE CHANGES ON THE MAIN PROGRAM

4.1 Most changes can be made without having an impact on the main program. However, due to the screen formatting constraints of PASCAL and the IBM 3278 terminal screen, some changes will require spacing changes within LISTING modules.

4.2 CHANGING THE COURSE NUMBER (col 4-7) - If there is an addition or deletion of a 3000 or 4000 level course, the spacing of LIST3 or LIST4 must be changed also. Adding a course will require the deletion of a trailing WRITELN within the module. Conversely, deletion of a course will require the addition of a trailing WRITELN. This will ensure proper screen format following the listing sequence.

4.3 CHANGING THE CURRICULUM CODE (col 13-14) - If there is an addition or deletion of an AS or CS curriculum code, the spacing of LISTAS or LISTCS must be changed also. Adding a course will require the deletion of a trailing WRITELN within the module. Conversely, deletion of a course will require the addition of a trailing WRITELN. This will ensure proper screen format following the listing sequence.

4.4 CHANGING THE REQUIRED CODE (col 48) - If there is an addition or deletion of a required course code, the spacing of LIST_REQUIRED_COURSES must be changed also. Adding a course will require the deletion of a trailing WRITELN within the module. Conversely, deletion of a course will require the addition of a trailing WRITELN. This will ensure proper screen format following the listing sequence.

4.5 CHANGING THE EMPHASIS AREA CODE (col 58,60,62 or 64)
If there is an addition or deletion of a emphasis area code, the spacing of LISTEMP must be changed also. Adding a course will require the deletion of a trailing WRITELN within the module. Conversely, deletion of a course will require the addition of a trailing WRITELN. This will ensure proper screen format following the listing sequence.

4.6 ADDING OR DELETING ANY COURSE -
If there is an addition or deletion of a course, the spacing of LISTALL and NAME must be changed also. Adding a course will require the deletion of a trailing WRITELN within the modules. Conversely, deletion of a course will require the addition of a trailing WRITELN. This will ensure proper screen format following the listing sequence.

ENCLOSURE 1

STRUCTURE OF "SKEDULER" DATA BASE

1) SET 1 - COURSE DATA (FIRST 57 ENTRIES)

COL(S)	CONTENTS	
1- 2	COURSE ID	
4- 7	COURSE NUMBER	
9	HOURS	
11	TERM ASSIGNED	(PRESET SCHEDULE) (= ACTUAL TERM PLUS 1) (1 = VALIDATED)
13-14	CURRICULUM	
16-17	PREREQUISITE #1 -	COURSE ID
19-23		COURSE NUMBER
24-25	PREREQUISITE #2 -	COURSE ID
27-30		COURSE NUMBER
32-33	PREREQUISITE #3 -	COURSE ID
35-38		COURSE NUMBER
40-41	PREREQUISITE #4 -	COURSE ID
43-46		COURSE NUMBER
48	REQUIRED COURSE	CODE (1 = YES / 0 = NO)
50	OFFERED IN THE:	FALL (1 = YES / 0 = NO)
52		WINTER (" / ")
54		SPRING (" / ")
56		SUMMER (" / ")
	EMPHASIS AREA COURSES:	
58	DSS	(2 = REQD / 1 = ELECTIVE / 0 = NA)
60	TS	(" / " / ")
62	ICN	(" / " / ")
64	CCNO	(" / " / ")

2) SET 2 - COURSE DESCRIPTIONS (SECOND 57 ENTRIES)
(EACH ENTRY CONSISTS OF A 10 LINE BLOCK)

COL(S)	CONTENTS
1-73	COURSE NUMBER, NAME, LECTURE/LAB HOURS, DESCRIPTION, PREREQUISITES

EXAMPLE DATA BASE ENTRIES:

SET 1

```
|...+....1....+....2....+....3....+....
IS 4185 4 0 AS MN 2155 MN 3105 OS 3101

                                    4....+....5....+....6....|
                                    IS 2000 0 0 1 1 0 2 0 1 1
```

SET 2

IS4185 - DECISION SUPPORT SYSTEMS ---------- (4-0)
The application and design of computer-based information systems for management planning, control and operations.
PREREQUISITES: MN2155, MN3105, OS3101, IS2000
*
*
* (this entry would include 6 blank lines
* because of the required blocking)
*
*

LIST OF REFERENCES

1. Keen, P. G. and Wagner, G. R., "DSS: An Executive Mind-Support System," Datamation, Vol. 25, No. 12, November 1979, pp. 117-122.

2. Alter, Steven, "A Taxonomy of Decision Support Systems," Sloan Management Review, Vol. 19, No. 1, Fall 1981, pp. 39-56.

3. Pressman, Roger S., Software Engineering, McGraw-Hill, Inc., 1982.

4. Deutsch, Michael S., Software Verification and Validation, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

5. Boehm, Barry W. Software Engineering Economics, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

BIBLIOGRAPHY

Alter, Steven, "A Taxonomy of Decision Support Systems," Sloan Management Review, Vol. 19, No. 1, Fall 1981, pp. 39-56.

Boehm, Barry W. Software Engineering Economics, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

Brooks, Frederick P.,Jr., The Mythical Man Month Addison-Wesley Publishing Company, Inc., 1975.

Deutsch, Michael S., Software Verification and Validation, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

Denise, Richard M., "Technology for the Executive Thinker," Datamation, Vol. 29, No. 6, pp. 207-216.

Graham, Neill, Introduction to Computer Science, West Publishing Co., 1982.

Hanson, Owen, Design of Computer Data Files, Computer Science Press, 1982.

Keen, Peter G. W., and Michael S. Scott Morton, Decision Support Systems: An Organizational Perspective, Reading MA: Addison-Wesley Publishing Company, 1978.

Keen, P. G. and Wagner, G. R., "DSS: An Executive Mind-Support System," Datamation, Vol. 25, No. 12, November 1979, pp. 117-122.

Keen, Peter G. W. "Decision Support Systems: Translating Analytic Techniques into Useful Tools," Sloan Management Review, Spring 1980, pp. 33-44.

Kroenke, David M., Business Computer Systems, Mitchell Publishing, Inc., Santa Cruz, Ca., 1981.

Kroenke, David M., Database Processing, Science Research Associates, 1983.

Loomis, Mary E. S., us:Data Management and File Processing, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.

Myers, Glenford J., The Art of Software Testing, John Wiley & Sons, Inc., 1978.

Myers, Glenford J., Reliable Software Through Composit Design, Van Nostrand Reinhold Company, Inc., 1975.

Naylor, Thomas H. "Decision Support Systems or Whatever Happened to M.I.S.?", Interfaces, August 1982, pp. 92-94.

Pressman, Roger S., Software Engineering, McGraw-Hill, Inc., 1982.

Rockart, John, "Chief Executives Define Their Own Data Needs," Harvard Business Review, Vol. 57, No. 2, March - April 1979, pp. 81-92.

SCHNEIDER, G. Michael, An Introduction to Programming and Problem Solving With Pascal, John Wiley & Sons, Inc., 1978.

Sprague, Ralph H., Jr., and Eric D. Carlson, Building Effective Decision Support Systems, Englewood Cliffs NJ: Prentice-Hall, Inc., 1982.

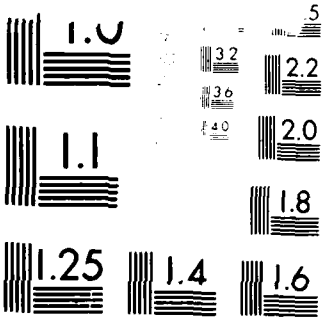
Wagner, G. R., "Optimizing Decision Support Systems," Datamation, Vol. 26, No. 5, May 1980, pp. 209-214.

Wagner, G. R. "Decision Support Systems: The Real Substance," Interfaces, April 1981, pp. 77-86.

Wagner, G. R. "Decision Support Systems: Computerized Mind Support for Executive Problems," Managerial Planning, September/October 1981, pp. 9-16.

AD-A168 843 A MECHANIZED DECISION SUPPORT SYSTEM FOR ACADEMIC SCHEDULING(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S M FENSTERMACHER MAR 86 3/3

UNCLASSIFIED F/G 5/9 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
.5
32
36
40

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor Norman R. Lyons, Code 54Lb Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor Michael P. Spencer, Code 54Sp Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	1
5.	Department Chairman, Code 54 Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	1
6.	Computer Technology Programs, Code 37 Naval Postgraduate School Monterey, California 93943-5000	1
7.	Commandant of the Marine Corps Code CCIS-42 Headquarters, United States Marine Corps Washington, D. C. 20380-0001 ATTN: CAPT S. M. Fenstermacher	1

END

DTIC

7-86